Manuscript Submitted to *Security and Safety,* For Peer Review Only, Vol. X, XXXXXXX (2026)
DOI: https://doi.org/10.1051/sands/2026020
sands.edpsciences.org

Article Type: Research

OPEN ACCESS

Section Category (Threat model)

The RipA Model: On the Domino Effect in Attacks against Informatization Architectures

Shiliang Ao*, Binxing Fang,

School of Cyberspace Science, Harbin Institute of Technology, Harbin, China.

Abstract

For informatization architectures with complex information interactions and dependencies, adversaries can exploit the relations among platforms and businesses that originally serve to maintain normal architecture operations, causing threats generated by attacks to transmit, spread, and amplify toward other components. We refer to this inherently unavoidable threat transmission pattern based on operational mechanisms as "threat ripple," and the resulting chain reactions of abnormal states as "ripple effects," namely, the domino effects formed by threat ripple. This paper discusses the formation and evolution mechanisms of such domino effects from a business oriented perspective. Traditional threat models cannot adequately describe this attack pattern.

Our paper designs an architecture model that integrates platforms and businesses for an informatization architecture, and explains the dependencies within it, as well as the attack methods through which an adversary can generate threat ripples. Based on this, we construct a Ripple-Affect(RipA) model to reflect how threats leverage internal dependencies to produce ripple effects, and corresponding mathematical representations are developed to support threat computation. Grounded in the architecture model, the proposed RipA model can characterize the domino effect of threat transmission within the architecture following an attack. By modelling a power production scenario, it is demonstrated that the proposed threat modelling approach can be used to evaluate threat events and support the quantitative analysis of threats.

Keywords: informatization architecture model, threat ripple, ripple effect, RipA Model, business dependencies, threat transmission, domino Effect

1 Introduction

APT organizations targeting critical infrastructure may leverage the domino effect caused by cyberattacks to transmit threats to targets that cannot be directly attacked. In the representative case of the Stuxnet event, although Iran's uranium centrifuge facility was a physical system rather than an information system, the worm manipulated the controllers of the centrifuges. As a result, the effect of erroneous network control operations transmitted across interconnected systems, ultimately turning a cyberattack into a physical attack that destroyed the centrifuge equipment [3][4].

* Corresponding author (email: shuaiasl@gmail.com)

This type of national infrastructure architecture, such as Iran's nuclear weapons development infrastructure, is an informatization architecture composed of multiple interconnected systems. It includes not only information systems that handle information-related tasks, but also informatization systems that rely on information systems to support non-information tasks [6]. Although these systems are independent in terms of management and/or operation, their interoperability and/or integrated combination often enable them to accomplish tasks that cannot be completed by a single system alone [40]. This paper approaches from the adversary's goals by decomposing the system into a "**platform component**" and a "**business component**," in order to characterize the domino effect of cyberattacks, particularly from the perspectives of vulnerabilities and dependencies in platform running and business operation.

1.1 Why Should Architecture Attacks Be Studied from a Business Perspective?

The operational mechanisms of an architecture depend on its businesses. However, a business is not an entity with a clearly defined physical form. Instead, it is a logical organizational arrangement formed around specific mission objectives to organize actions, allocate resources, and constrain platform running processes. Even if all physical assets that may be subject to cyber attacks in real scenarios are taken into account, the business remains difficult to capture. To investigate the operational mechanisms of an architecture, abstract modeling is required to make businesses explicit. Businesses and platforms must both be represented as modeling objects that exhibit dependencies and state changes and support the transmission of effects. Otherwise, the analysis may remain limited to physical objects such as hardware, systems, modules, and interfaces.

On one hand, businesses determine the actual significance and ultimate consequences of attack effects. An attack on the same platform may produce entirely different consequences in different business processes. For example, a database anomaly in an ordinary test system may cause only data errors. However, if the database supports power dispatch, industrial control decisions, or financial clearing, the same anomaly may disrupt the entire architecture. The security problems of a platform do not depend solely on its own vulnerabilities or the severity of its operational anomalies. The businesses it supports, its position within an operational chain, and the missions it is associated with jointly determine the ultimate consequences of an attack. For architecture operators, it is necessary to go beyond merely identifying attack techniques and vulnerabilities and further determine which businesses may be affected by local security problems, whether mission execution may be hindered, and at which critical points threat transmission should be interrupted to reduce the overall effects on the architecture.

On the other hand, business relations can reveal the actual significance behind platform interactions. Many relations among platforms within an architecture are not inherently meaningful. Only when they are embedded in specific business processes and operational logic can we understand why these relations exist and what consequences may arise when abnormal states are transmitted through them. From a technical perspective, connections between platforms may merely involve interface calls, data transmission, or permission management. From a business perspective, however, these interactions essentially exist to support business operations and mission accomplishment. For

example, although the IoT monitoring nodes, databases, middleware hosts, and reaction furnaces in a steel plant appear to exchange information across different platforms [64], only by placing these interactions within a business chain involving production monitoring, state feedback, reaction parameter adjustment, and production execution can we understand why an abnormality in a low privilege and easily overlooked monitoring node may still transmit its effects through subsequent business processes, thereby affecting production control and ultimately leading to a production accident.

Therefore, platforms carry running logic oriented toward machine execution and system processing and serve business operations, whereas businesses carry operational logic oriented toward achieving architecture objectives and accomplishing missions, thereby giving platform interactions actual significance at the architecture level. Threat modeling from a business perspective enables defenders to view attacks from the adversary's perspective and examine why a specific component is selected as a target, what the actual goal of the attack is, and how a local security problem produces effects across the architecture.

1.2 Why do attacks generate domino effects?

In our previous work [66], we proposed the threat ripple model for information architecture scenarios, as shown in Figure 1, and introduced the concept of threat ripple in cyberspace for the first time. On this basis, this paper further extends the concept to informatization architectures scenarios.

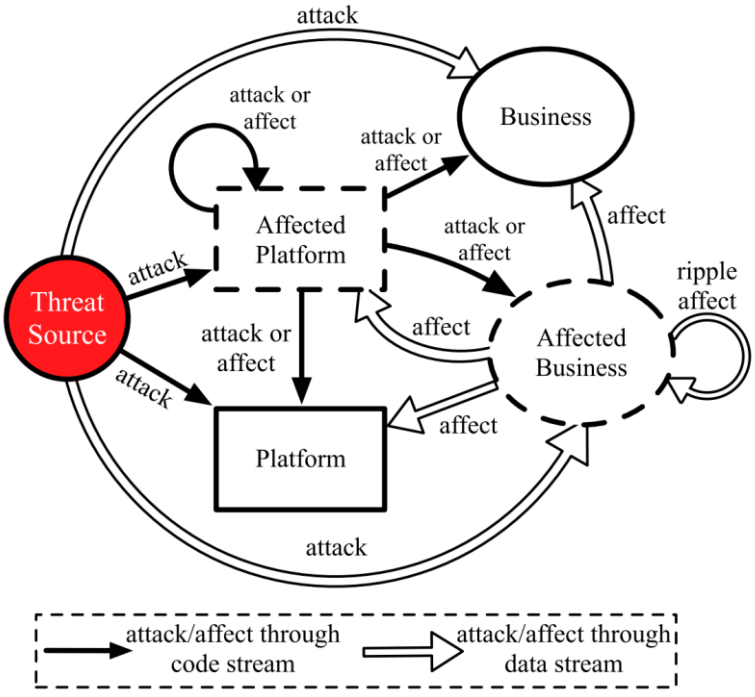


Figure 1 Threat ripple model

In an informatization architecture, different businesses and platforms are connected through complex dependencies and interactions, making the states of different components interrelated. When an attack causes a component to become abnormal, the resulting threat can utilize these relations, originally intended to maintain

normal architecture operations, to transmit, spread, and amplify toward other components. We refer to this threat transmission pattern based on operational mechanisms as **threat ripple**, and the resulting chain reactions of abnormal states as **ripple effects**, namely, the domino effects formed by threat ripple.

The unavoidability of threat ripple lies in the fact that threat ripple does not rely on additionally constructed attack paths, but is carried by the operational mechanisms that maintain the normal state of an architecture. Adversaries can utilize these relations, which are originally intended to maintain normal architecture operations, to transmit and amplify the effects of attacks across platforms and businesses. As long as dependencies and interactions exist, abnormal effects may naturally spread along with normal operational processes.

Although there are many studies on threat modeling [5][8][26][27][28][29], they generally lack analysis of the effect of attacks on business operations and do not couple attack mechanisms with the operational mechanisms of the target architecture to examine the process of threat spreading. As a result, they are unable to fully characterize the ripple effects generated by threats through state transitions within the architecture after it is subjected to an attack.

1.3 How does the attack affect the target component?

When the target component cannot be attacked directly, an adversary may instead attack the components on which the target depends, either directly or indirectly, to achieve the intended effect. In this paper, a component that begins to generate and propagate threats after being attacked is referred to as a threat transmission origin component. A component that transmits threats between the origin and the target is referred to as a threat transmission intermediate component.

1.3.1 Utilize direct dependency

For relationships in which the target component directly depends on the origin component, this paper classifies them into two types based on the ownership of the underlying resources. One type, referred to as **support dependency**, occurs when the origin component has certain resources and provides them to the target component to ensure it performs normal. If an adversary can maliciously manipulate the origin component or cause it to enter an abnormal state, the mechanism by which it supports the target component can be affected. For example, after the information system of the Croatian public hospital KBC Zagreb was attacked, the medical records business it supported “reverted 50 years, relying only on paper and pencil” [37].

Another type, referred to as **management dependency**, occurs when the target component has the resources but relies on the origin component for security management to ensure the proper use and planning of those resources. Similarly, an adversary who compromises the origin component can achieve their objectives by manipulating, or by causing the origin component to enter an abnormal state and thereby abusing, its management mechanism over the target component. For example, in the cyberattack on the water system in the US city of Oldsmar, the hacker accessed SCADA control system remotely and attempted to increase the sodium hydroxide concentration to dangerous levels [29].

1.3.2 Utilize indirect dependency and composite dependency

The limitation of attacks that utilize direct dependency relationships lies in what can be described as a “zero-or-nothing rule”: Manipulating the origin component typically requires a high level of privilege and is

therefore difficult to achieve. Moreover, only when the origin component is successfully driven into an abnormal state will the target be affected; otherwise, the attack will fail. In an informatization architecture with complex dependency relationships, the components on which a critical target directly depends are often heavily protected and difficult to attack. However, these components may lie within other dependency chains, whose upstream parts may contain weak components, and the normal state of the target component can be indirectly affected by the states of them.

This provides the adversary with more actionable entry points and opportunities: when a target component cannot be attacked directly, or its direct dependency relationships cannot be utilized, the adversary can instead look for weak components on which it indirectly depends, or with which it has state dependency relations. By driving these components into abnormal states and thereby disrupting their effective capability output, ripple effects can be generated that ultimately transmit to the target component. For example, after the Belgian oil terminal operator SEA TANK was subjected to ransom attack, its paralyzed systems were unable to support barge operations, indirectly forcing oil tankers to wait outside the port and disrupting both oil loading and transshipment activities [9].

1.4 Threat transmission mechanisms after an attack

The essence of the threat ripple is that threats utilize dependencies to transmit abnormal states. Therefore, the rippling of threat is conditional: One is that the state of the platform or business component does not necessarily being affected and become abnormal when it is exposed to a threat, but only if the input threat meets certain conditions. For example, in the Lebanon pager explosion event, the devices could be used normally under ordinary conditions, but the explosives were triggered upon receiving a specific signal [18].

Another is that unless the abnormal state meets certain conditions, affected component does not necessarily transmit threats outward. For example, in a backup system, files tampered with by hackers will transmit the threat to the operational system only after the host performs an initial system restoration [13]. Furthermore, even when an affected component satisfies the conditions required to transmit a threat to one dependent component, it may not satisfy the conditions needed to transmit the threat to others. This is because the enabling types and modes provided by a component to its individual dependents may be different. For instance, the XMRig malware developed for macOS cannot be executed on a Windows host even if it is downloaded [16].

The mechanics of threat transmission are actually the architecture's own operational mechanics: with no change in the architecture environment, the attributes of platform and business components, as well as the dependencies among them, are determined by the architecture's design, inherent characteristics, and management patterns. The adversary does not need to understand all the dependency logic in the architecture, as threats will transmit automatically along dependency chains as long as the state transmit conditions are satisfied. This enables the effects of an attack to spread rapidly throughout the architecture. Even if the component directly attacked is quickly repaired, the resulting domino effects are difficult to eliminate in a short period of time.

Although many attack graph models have studied the impact of threats [7][10], they do not consider the effects of threats on business states. Some papers also describe threat models that can be constructed in architectural scenarios and characterize the logic of threat transmit [2][14][12][23], none of them take into account the conditions for threat transmission.

1.5 Express the domino effect of threat transmissions

This paper designs an informatization architecture model to characterize the constraints and interaction patterns among platform and business components within the architecture. Based on this model, the functions of direct, indirect, and state dependencies in the threat ripple process are explained. Furthermore, the paper constructs the RipA model, which couples the adversary's attack logic with the architecture's operational logic to depict the logics and effects of threat transmissions within the architecture. Additionally, mathematical expressions are designed for the RipA model to support the calculation and analysis of threat transmission conditions and effects on component states, thereby characterize the domino effect of threat transmission. Finally, the paper models a typical power production architecture as a case study.

Similarly, for attacks targeting a non-information architecture, adversaries can generate cross-architecture ripple effects by attacking external information architecture that provide capabilities to it, thereby achieving their goals. The modelling approach in this paper is able to characterize such pattern of attacks.

Our contributions:

1.informatization architecture model is constructed: this paper proposes an architecture model capable of describing the information interactions among information systems and non-information systems from a business perspective. This facilitates a better understanding of operational mechanisms both within and cross architectures, thereby building endogenous security during the process of architecture design, operation and maintenance.

2.Expanded expression of threat ripple: combined with the attack methods capable of generating ripple effects, this work systematically explains the domino effect of threat transmission. This extends the scope of threat ripple characterization from information architectures to informatization architectures that include non-information business.

3.Presented the RipA model that expresses the domino effect in attacks: this helps defenders identify the logic through which risks and threats can spread and amplify their effects, and establish environments for offensive-defensive inference. Even when an attack has already occurred, defenders can use the RipA model to promptly block the ripple effects of threats within the architecture.

4.Developed the mathematical representation for threat ripples: this paper provides a formal representation of the conditions, logic, and resulting component state changes involved in the process by which threats transmission and cause effects within an architecture, thereby establishing a theoretical foundation for the analysis and quantification of ripple effects.

5.Proposed a weakness analysis methodology for Informatization architecture: our modeling approach can determine which information platforms or businesses, once attacked, are likely to trigger the spread of effects. This can help the defenders to identify the key targets of the adversary and thus focus protection to them.

The remainder of this paper is organized as follows. In section 2, related work about ripple effect and threat model is discussed. In section 3, the informatization architecture model is constructed to characterize the information interactions and dependencies among components, and discusses the patterns through which attacks utilize to transmit threats. Section 4 introduces attack mapping relations and threat-affect mapping relations, then proposes the RipA model based on the informatization architecture model. In section 5, the mathematical representations of threat ripple and ripple effects, together with the mathematical formulations and description framework of the models, are presented. Section 6 characterizes a typical threat scenario utilizing threat modeling approach proposed in this paper. Section 7 compares the RipA model with other representative threat

models. Finally, section 8 and section 9 provide the discussion and conclusion of this research, respectively. The main mathematical symbols used in this paper are summarized in Table 1.

Table 1 Nomenclature of mathematical symbols

Symbol	Meaning
$\bigcup$	Union of sets
$\in$	An element belongs to a set
$\wedge$	Logical AND
$\bullet$	Vector condition judgment symbol. The condition is considered satisfied when the specified indicators meet the corresponding conditions.
$\models$	Satisfies a constraint
$\odot$	Hadamard product
$\rightarrow$	Mapping relation
$\mapsto$	Corresponding result under a predefined function or rule
$\cdot \mid x$	Indicates that only x can be obtained or is concerned under the current condition
${}_{A}\vert_{B}^{R}$	Transmission symbol. Indicates a unidirectional transmission relation R from A to B
${}_{A}\vert_{B}^{(n)}$	Depth symbol. Indicates that the unidirectional transmission depth from A to B is n
$< \mathbf{C}_t >$	Sequence endpoint marker. Indicates that the current sequence ends at $\mathbf{C}_t$
$\mathbb{R}^{m\times n}$	$m\times n$ dimensional real matrix space
∂	Partial derivative symbol

2 Related works

The ripple effect has not been specifically discussed in cybersecurity, and there is also a lack of a complete theoretical account of how the effects caused by an attack spread and become amplified by utilizing the internal relations within a target. At present, apart from the business oriented threat model proposed in the authors' previous work [66], no other model is capable of characterizing the threat ripple process. Against this background, this paper reviews research adjacent to this problem to clarify the research gap that it seeks to fill.

2.1 Researches on ripple effect

When a system is subjected to an initial disturbance, the resulting impact may transmit outward and affect an increasingly large portion of the system. This phenomenon is known as the ripple effect, analogous to the waves generated by an object falling into water and continuously spreading outward. It differs from the butterfly

effect, which emphasizes that small initial changes may be significantly amplified through nonlinear evolution, ultimately producing substantial and unpredictable consequences [39].

In macroeconomics, the ripple effect is used to describe multiplier effects. For example, from a global perspective, monetary policy decisions, particularly those made by major economies such as the United States, may generate ripple effects throughout the global economy [68]. In sociology, the ripple effect refers to the extension of social influence beyond the context in which the initial interaction occurs. Examples include how population growth, urbanization, and other factors affect coastal resources, how environmental degradation affects people's lives [69], and how the influence of an individual or event spreads through social networks from immediate neighbors to second-order and more distant neighbors [71]. In the cultural domain, the study in [72] shows that individuals from East Asian cultural backgrounds are more inclined than those from Western cultural backgrounds to consider the indirect, long-term, and complex consequences of events.

In supply chain security engineering, ripple effect researches focuse on the spread of disruptions across supply chain networks and their consequences for resilience and operational performance, such as paper [70]. Paper [73] analyzes the ripple effect resulting from the transmission of supply chain disruptions through the network, examines its impacts on supply chain performance, structural design, and planning parameters, then distinguishes it from the bullwhip effect. The "Handbook of Ripple Effects in the Supply Chain" investigates cascading effects in supply chains from a multidisciplinary perspective and introduces risk-mitigation techniques for industrial and service supply chains, supported by real-world cases from both sectors [11].

However, the ripple effect has not yet been specifically examined in the field of cybersecurity, and a comprehensive theoretical account of how attack effects spread and amplify through the internal dependencies of a target is still lacking. To address this gap, this paper proposes the concept of threat ripple, systematically characterizes domino effects in cyberspace in terms of their formation mechanisms, transmission conditions, and effect patterns, and further develops the corresponding theoretical framework and modeling approach.

2.2 Researches on threat model

Existing threat modeling approaches can be categorized from two perspectives: asset & impact oriented, and adversary & threat oriented, respectively.

2.2.1 Asset & Impact-orientated approach

Among widely used international security models, the ONDI Cyber Threat Framework [12] and the Attack Tree Model [14] provide clear methods for describing threats faced by cyber asset architectures. In terms of security design, STIX offers a method to incorporate structured threat intelligence into specific instances [23]. The Open Group Architecture Framework (TOGAF) [24] and the Department of Defense Architecture Framework (DODAF) [25] provide methodologies for the design, planning, implementation, and management of technical architectures for complex systems. For threat and risk assessment, OWASP [21] and PASTA [22] enable knowledge-oriented threat modeling. Models like STRIDE & DREAD [15] and OCTAVE [17] are effective in supporting design analysis and testing. However, these approaches mainly take platform assets as the core modeling objects, lacking systematic consideration of business operational states and their security effects.

Although some studies have considered the impacts caused by attacks—for example, the Cybersecurity Risk Management (CSRM) method proposed in [28] includes qualitative and semi-quantitative analysis, the method in [26] proposes a method to measure the impact of malicious events on complex systems, and the

approach in [27] utilizing security principles to identify and analyze critical assets and their potential vulnerabilities—they all have difficulty characterizing the mechanisms by which threats spread outward after assets are affected. In contrast, our RipA model can not only express the effect of threats on business but also characterize the threats transmission logic among platform and business components.

2.2.2 Adversary & Threat-orientated approach

The most representative cyber kill chain model divides the lifecycle of an attack into seven stages [1]. The MITRE ATT&CK framework [2], CAPEC™ [19], and MITRE's TARA [20] provide resources for threat modeling from the perspectives of tactical, technical, and process-oriented (TTP). Some academic studies have also explored attack behaviors and security assessment [8][31][5], but none of these models consider the effects of threats at the business level.

Paper [30] proposes a threat assessment method for enterprise networks. To estimate the likelihood of exercising threats, [33] used a quantitative approach to evaluate the motivation, capabilities and opportunity of attackers, and [45] presents theoretical and practical approaches for cyber risk management based on the cyber kill chain. Although these studies consider both adversary behaviors and attack consequences, none of them conduct in-depth analysis of the subsequent effects of threat spreading. Our modelling approach is able to express the attack pattern of the adversary utilizing the internal operating mechanism of the target architecture to amplify the attack effects.

3 Informatization Architecture Model

This paper proposes an Informatization architecture model, to characterize the information interactions, mission processing modes and dependences in the Informatization architecture.

3.1 Model Representation

We abstract the Informatization architecture into sets of the following three constituent components, of which:

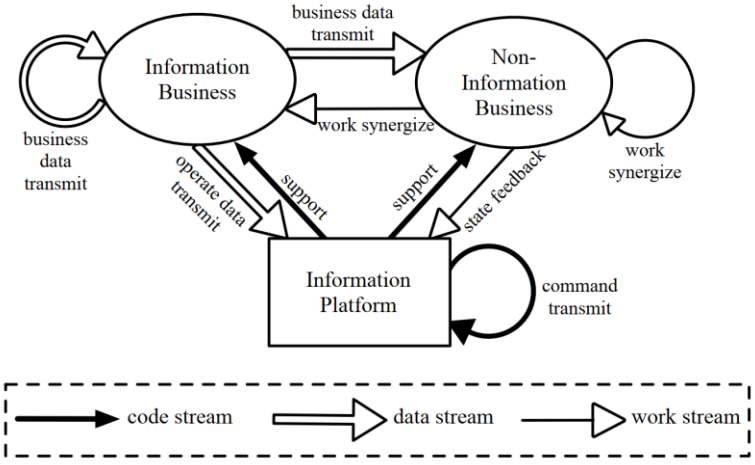


Figure 2 Informatization architecture model

Information Platform is defined as the infrastructure running the code data that supports the business, expressed in solid rectangle[1] **Information business** is defined as the service function that the Informatization system provides to the outside world through business data, whereas **non-information business** is defined as the service function that the Informatization system provides to the outside world in a non-data form, and both of them are expressed in a solid ellipse. The granularity of these components can be determined in practice according to the needs of the scenario.

The relations between information platforms and information businesses are established through data interactions. In this paper, such data are abstracted into two categories: **code data** refer to sequences of instructions or command symbols written in programming languages that can be understood and executed by computers to define, trigger, or modify the running logic of a platform. They may take the form of program code, configuration rules, instructions, and other executable content used for platform running. **Business data** refer to data generated, received, processed, or transmitted during service provision or work activities. They may also be collected, processed, computed, or generated by a platform in response to business inputs and are used to support business activities. When a business requires a platform to perform a specific task, the operational instructions or execution rules submitted to the platform are also regarded as business data because their origin and semantics remain part of the business activity. Such data constitute a category of business data known as **operation data**, which are used to drive the platform in performing tasks and support business operations. During transmission, the two types of data form code data streams, hereinafter referred to as code streams, and business data streams, hereinafter referred to as data streams, respectively.

In real-world scenarios, advances in the encapsulation of code and data may cause the same object to exhibit different properties in different contexts, making the boundary between code behavior and data behavior increasingly blurred. Traditionally, business operations were generally driven by platforms executing specific code logic. However, as code becomes increasingly reusable and general-purpose, a growing number of business functions are driven by data inputs and data configurations. For ease of discussion, this paper no longer distinguishes between code and data solely according to whether the object itself takes the form of code or data. Instead, the distinction is made according to the object on which the information acts and the purpose it serves, with a focus on whether the information primarily serves the platform or the business: information that primarily serves the platform by defining, triggering, or modifying its running logic is regarded as code behavior or a code stream. Information that primarily serves the business by expressing business semantics, supporting business collaboration, or driving business tasks is regarded as data behavior or a data stream.

For non-information business, we abstract its external behavior during operating as business work, which forms a business work stream when interacting with other component - hereafter referred to as **work stream**. In some scenarios, the work stream may be accompanied by the data stream. The Informatization architecture

[1] This paper does not discuss non-information platforms that is not considered part of the cyberspace. All platforms including cyber physical systems in this paper refer to information platforms.

model is shown in Figure 2, there are six corresponding binary relations:

Command transmit: Commands are transmitted from platform to platform through code stream to achieve functional synergy among platforms.

Business data transmit: Information is transmitted through business stream to achieve capability synergy. In addition, the information business will provide data to the non-information business to cooperatively complete the mission.

Support: The platform supports its corresponding business through running code. This paper considers this as a special form of code transmission from the platform to the business.

Operate data transmit: Business uses its corresponding platform through entering operation data into the platform.

State feedback: The physical facility used for non-information business operating provides feedback on its current state to the cyber-physical system (CPS) platform, such as industrial control system, to help CPS to ensure the non-information business complete the mission.

Work synergize: Non-information businesses interact with each other through work streams to achieve synergized capabilities. They also interact with information businesses through work streams to complete missions. Although this relationship has been labelled in the model, synergistic work that does not involve information attributes is not considered part of the cyberspace and is therefore, not discussed in detail in this paper.

These relationships describe the patterns of information interaction and capability coordination among platform and business components, forming the basis for subsequent threat transmission analysis. Based on the above model and relationship definitions, adversaries can leverage these dependencies to transmit threats from origin components to target components. The next section will classify and explain t these leverage methods.

3.2. Attacks utilize dependency

The ripple effect that occurs after an informatization architecture is attacked is, in fact, the process by which threats spread and take effects leverage complex dependencies. When a target component cannot be attacked directly, the adversary can attack a weak component instead, utilizing the dependencies among components to transmit threat to the target component.

3.2.1 Attacks utilize direct dependency

When performing a specific mission, for the direct dependencies between components, we classify them into two types according to the ownership of resources. One is that the target component has the required resources but lacks the ability to complete the mission independently. It requires the origin component to provide proper management to ensure that it can accomplish the mission safely. This dependency is referred to as the **management dependency** of the target component on the origin component.

And the other is, the target component lacks the necessary resources and requires the origin component to provide the proper support to ensure that the mission can be safely completed. This dependency is referred to as the **support dependency** of the target component on the origin component. In neither of these relations, the target components have the capability to maintain its own normal state independently.

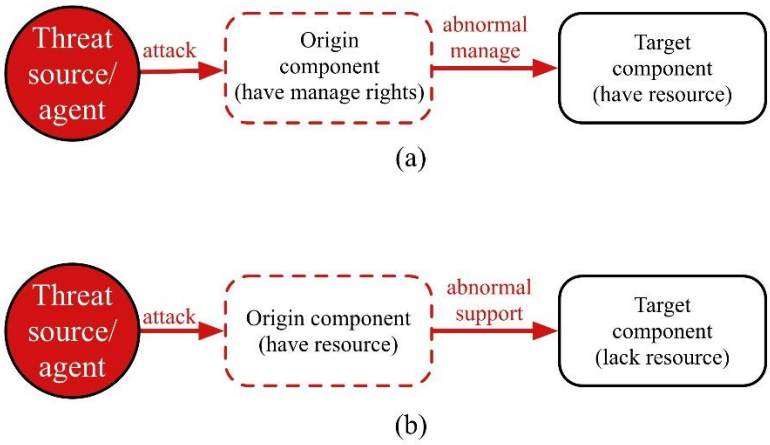


Figure 3. Attacks utilizing direct dependencies. (a) Management dependency. (b) Support dependency.

The attack patterns that utilize direct management dependency and direct support dependency are illustrated in Figure 3(a) and Figure 3(b), respectively. An adversary may attack the origin component and manipulate its management actions or resource support (typically requiring elevated privileges), or force it into a specific abnormal state, thereby causing it to exert abnormal management or abnormal support toward the target component, thereby achieving the goal of transmitting threats to the target.

3.2.2 Attacks utilize indirect dependency

Indirect dependencies are, in fact, extensions of direct dependencies. If two components with a management dependency or support dependency are not directly connected, but instead establish indirect dependency through intermediate component(s), then they are referred to as **indirect management dependency** and **indirect support dependency**, respectively. In both of them, the intermediate components establish collaborative relationships with both the origin component and the target component, and are responsible for forwarding or relaying the received business streams or code streams, or manufacturing them as required before transmitting the generated results to downstream components. Therefore, the state of the origin component not only determines the state of the target component, but may also, under specific conditions, determine the state of intermediate components.

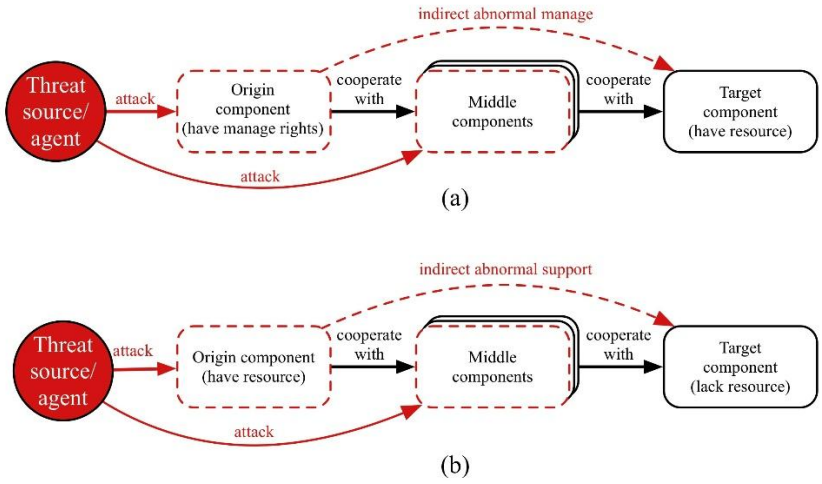


Figure 4. Attacks utilizing indirect dependencies. (a) Indirect management dependency. (b) Indirect support dependency.

Attack patterns that utilize indirect management dependency and indirect support dependency are illustrated in Figure 4(a) and Figure 4(b), respectively. In addition to attacking the origin component to enable threats to reach the target component by penetrating intermediate components, an adversary may also attack any intermediate component, manipulate the coordination it provides to downstream components, or drive it into an abnormal state so as to affect the existing indirect management or support logic. These enable threat transmission to the target component, potentially intensifying and amplifying the effects of the attack. The specific threat transmission logic is determined by the architecture's internal operating logic.

It should be noted that attacks on intermediate component(s) essentially utilize the dependency between the origin component and the target component. For the component that is not in a dependency chain, driving its state abnormal does not necessarily affect the components it connects to. This depends on the specific scenario and coordination mode, and cannot be regarded as a sufficient condition for threat transmission, thus is not discussed in this paper.

3.2.3 Attacks utilize state dependency

In practical scenarios, multiple support dependencies or management dependencies within an architecture may interconnect or intersect, thereby forming composite dependencies. If the target component C_t forms a composite dependency with the origin component C_s through the cooperation of $C_{s+1}, C_{s+2}, \ldots C_{t-1}$, then its normal state will depend on that of C_s: once the state of C_s changes, the support or management patterns between it and the downstream collaborating components will also change accordingly, thereby forming a state transmission that process that ultimately affects the state of C_t. This type of dependency is referred to as the target component's **state dependency** on the origin component. State dependency is a combination of multiple support dependencies and/or management dependencies. The cooperation link between C_s and C_t may jointly consist of multiple code streams, business streams and work interactions.

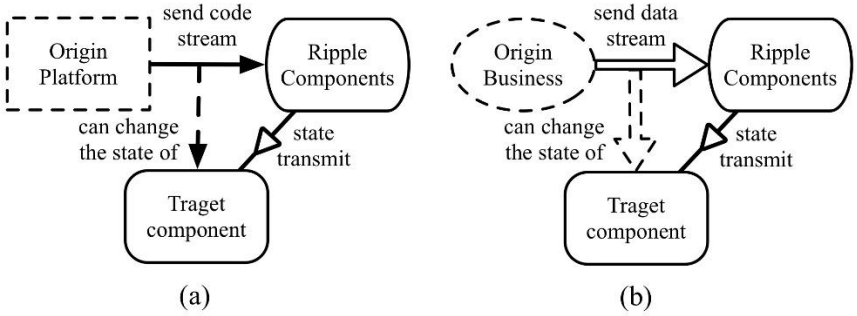


Figure 5. State dependencies. (a) Origin platform affects the target component through state transmission. (b) Origin business affects the target component through state transmission. Rounded rectangles denote that the

object may be either a business or a platform component.

In order to describe the state transmission process, we introduce the T- shape symbol, and use a drum-shape diagram to represent the state dependency involving multiple intermediate components, as shown in Figure 5(a) and 5(b). Where the solid horizontal line represents the direct-action relationship, meaning that the behavior of the origin component directly acts on connected intermediate component. The dotted vertical line represents the effect result, that is, the change in the state of the target component is essentially caused by the state change in the origin platform or the origin business. Therefore, the indirect dependency can be regarded as a single interaction mode state dependency, whereas the direct dependency is the minimal state dependencies without any intermediate component.

For the adversary, if the origin component C_s can be attacked and driven into an abnormal state, the consequence will first act on C_{s+1}, subsequently, the abnormal state will transmit step by step along the dependency sequence to C_{t-1}, and ultimately affects the target component C_t, as shown in Figure 6. The notation $C_{s+1} \mid C_{t-1}$ represents the set of intermediate components that carry the abnormal state transmission of abnormal states, i.e., the components from the $(s+1)$ th to the $(t-1)$ th in the dependency chain.

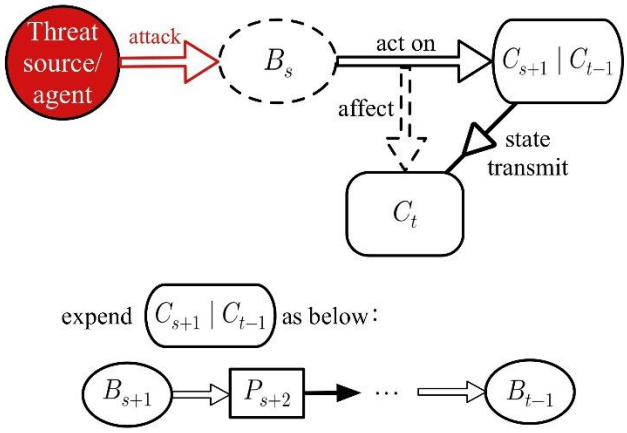


Figure 6 Attack utilizes state dependency.

Meanwhile, we define the length of the dependency chain affected once C_s becomes abnormal as the **ripple depth** m. In Figure 6 the ripple depth of C_s is $m = t - s$. In real scenarios, the state transmission logic is determined by the architecture's internal mechanisms, therefore, different abnormal states of C_s may correspond to different ripple depths, i.e., $m \in [0, t-s]$. Threat transmission is, of course, conditional, and the specific constraints will be explained in detail in section 5.

During the process in which threat spreads in the form of abnormal state transmission, the states of intermediate components may change, but do not necessarily become abnormal. This depends on the adversary's tactical design and the running or operational mechanisms of the intermediate components themselves: if the

threat transmits by leveraging the components' inherent mechanisms, the components' state may change but remain normal. If the threat transmits by maliciously disrupting or tampering with the components' mechanisms, then the components will exhibit abnormal states.

4 RipA Model

This paper constructs the RipA model based on the informatization architecture model to characterize the threat transmission mechanism of dependency-based attacks.

4.1. Objects Definition

This paper abstracts threat actions that lead to state changes in platforms and businesses into two types: attack and affect, and threat data into two types: threat code data and threat business data:

The **attack** is defined as the action of changing the business state through data streams, or the platform state through code streams, where streams are with attack attributes.

The **affect** is defined as without any new attack code or attack data, but due to the component's own running state or operating state being affected, its associated platform follows the normal logic of interacting with it in code, as well as associated business follows the normal logic of interacting with it in data, will cause the abnormal state changes of the associated platform or business.

Threat code data is defined as input code that can change the normal running state of the platform, or platform's running code that can change the normal operating state of its supporting business.

Threat business data is defined as input business data that can change the normal operating state of the business, or the input operational data that can change the normal running state of the platform.

In real-world scenarios, threats are generated after platforms and businesses are attacked or affected, and transmit among components through data streams. The logic of transmissions is determined by the architecture's own mechanisms, so the target of the adversary and the final destination of the threat ripples may be different. For example, in the Ukrainian blackout, the adversary's goal was achieved when power business was paralyzed, but the effects of the blackout were still spreading [34]. For ease of modeling, this paper considers the adversary's target to be the same as the final affected destination. To better describe the effects of threat ripples on the informatization architecture, the target businesses in the examples below are all non-information businesses.

We define the RipA model D:

$$D = \{C, H, F\} \tag{1}$$

where C is defined as the set of all objects in model D, H is the set of attack functions, and F is the set of threat-affect functions. For C:

$$\begin{aligned} C = \{&\textit{Threat source}, \textit{Affected Business}, \textit{Target Business}, \\ &\textit{Affected Platform}, \textit{Target Platform}, \textit{Ripple Components}\} \end{aligned} \tag{2}$$

For the objects of the threat's action, there exists subsets: $C_a, C_r, C_t \subset C$, where:

C_a is the set of abnormal objects in the architecture that initiate the threat transmission after being

attacked, i.e.:

$$C_a = \{ Affected\ Business,\ Affected\ Platform \} \tag{3}$$

C_t is the set of abnormal objects, i.e.:

$$C_t = \{ Target\ Business,\ Target\ Platform \} \tag{4}$$

C_r is the set of intermediate business components or platform components involved in the state transmission process., i.e.:

$$C_r = \{ Ripple\ Components \} \tag{5}$$

For threat actions among the objects in D, we define them as two types of mapping relations: attack mapping relations and threat-affect mapping relations.

4.2 Attack mapping relations

Set H includes two types of attack relations in which the threat source or threat agent targets on the business and the platform:

$$H = \{h_{data}, h_{code}\} \tag{6}$$

Here, h_{data} represents the attacks executed through business stream, and h_{code} represents attacks executed through code stream. They are illustrated in Figure 7(a) and 7(b), respectively:

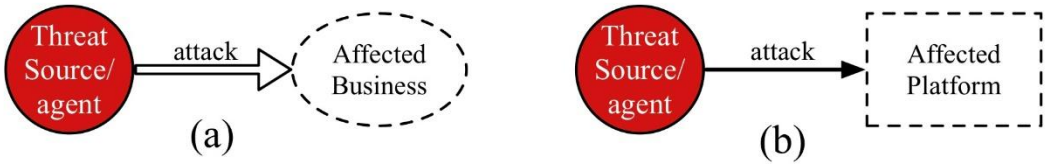


Figure 7. Attack mapping relations. (a) Attack though data stream. (b) Attack through code stream.

h_{data} : $Threat\ source \rightarrow Affected\ business$: The adversary launches an attack against the business to cause abnormality in its operational state. E.g., Fraudsters use stolen credit card information to make overpayments on products, then request the excess amount to be refunded to another account, thereby converting available credit balances into cash [49].

h_{code} : $Threat\ source \rightarrow Affected\ platform$: The adversary launches an attack against the platform to cause abnormality in its running state. E.g., attackers exploited the MS17-010 Eternal Blue vulnerability in Windows to launch the WannaCry ransom attack. As a result, a large number of computers and servers around the world failed to run normally [35].

4.3 Threat-affect mapping relations

The set F contains three mapping relations, which are used to represent the **management-affect relation** and **support-affect relation** of C_a directly acting on C_t , as well as the **state-affect relation** that C_a exerts on C_t through C_r :

$$F = \{F^{mg}, F^{sp}, F^{st}\} \tag{7}$$

4.3.1 Management-affect relations

F^{mg} represents management-affect relations of C_a on C_t :

$$F^{mg} = \{f^{mg}_{data}, f^{mg}_{code}\} \tag{8}$$

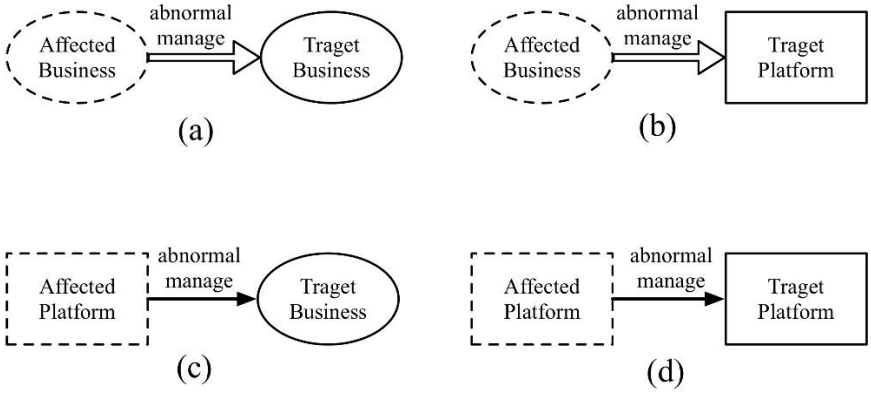


Figure 8. Management-affect relations. (a) and (b) represent abnormal management through data streams, (c) and (d) represent that through code streams.

Here, f^{mg}_{data} represents the management effect caused by the transmission of threat business data. The mapping relation is expressed as:

$$f^{mg}_{data} : Affected\ business \rightarrow C_t \tag{9}$$

Expending Eq.9, as shown in Figure 8(a) and 8(b), respectively:

$f^{mg}_{data} : Affected\ business \rightarrow Target\ business$: The state affected business performs abnormal management on the target business. E.g., In 2012, wireless interference from portable 3G routers used by passengers disrupted normal communications in the Shenzhen Metro signaling system, causing multiple trains to trigger frequent emergency braking during operation and affecting metro services [38].

$f^{mg}_{data} : Affected\ business \rightarrow Target\ platform$: The state affected business performs abnormal

management on the target platform. E.g., while troubleshooting an issue with the S3 (Simple Storage Service) billing system, an AWS engineer entered an incorrect command, causing more servers than intended to be removed [67].

And f_{code}^{mg} represents the management effect caused by the transmission of threat code data. The mapping relation is expressed as:

$$f_{code}^{mg} : Affected\ platform \rightarrow C_t \tag{10}$$

Expending Eq.10, as shown in Figure 8(c) and 8(d), respectively:

$f_{code}^{mg} : Affected\ platform \rightarrow Target\ business$: The state affected platform performs abnormal management on the target business[2]. E.g., vulnerabilities in Rockwell's programmable logic controllers (PLCs) and engineering workstation software can be remotely exploited by attackers, thereby modifying automated business processes [41].

$f_{code}^{mg} : Affected\ platform \rightarrow Target\ platform$: The state affected platform performs abnormal management on the target platform. E.g., After the port management system of Nagoya Port in Japan was compromised by the LockBit team, the hosts and printers under its control were subsequently encrypted, ultimately causing the port's container scheduling system NUTS to halt running [48].

4.3.2 Support-affect relations

F^{sp} represents support-affect relations of C_a on C_t:

$$F^{sp} = \{f_{data}^{sp}, f_{code}^{sp}\} \tag{11}$$

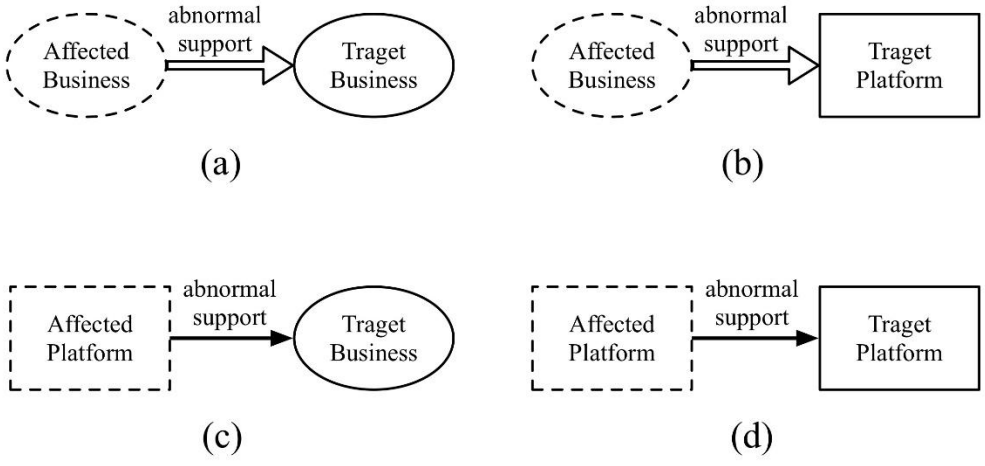


Figure 9. Support-affect relations. (a) and (b) represent abnormal support through data streams, (c) and (d)

[2] In fact, the business became abnormal due to using abnormal data generated by the running platform.

represent that through code streams.

Here, f_{data}^{sp} represents the support effect caused by the transmission of threat business data. The mapping relation is expressed as:

$$f_{data}^{sp} : Affected\ business \rightarrow C_t \tag{12}$$

Expending Eq.12, as shown in Figure 9(a) and 9(b), respectively:

$f_{data}^{sp} : Affected\ business \rightarrow Target\ business$: The state affected business provide abnormal support to the target business. E.g., DDoS attacks launched by a mobile phone botnet against 911 emergency services posed a serious threat to service availability, preventing normal emergency response operations from being carried out [50].

$f_{data}^{sp} : Affected\ business \rightarrow Target\ platform$: The state affected business provide abnormal support to the target platform. E.g., The WireX botnet malware, disguised as a legitimate Android application, was successfully listed on the Google Play Store and widely downloaded and used. The attackers then gained access to a large number of infected enterprise users' Android devices [47].

And f_{code}^{sp} represents the support effect caused by the transmission of threat code data. The mapping relation is expressed as:

$$f_{code}^{sp} : Affected\ platform \rightarrow C_t \tag{13}$$

Expending Eq.13, as shown in Figure 9(c) and 9(d), respectively:

$f_{cp}^{sp} : Affected\ platform \rightarrow Target\ business$: The state affected platform provide abnormal support to the target business. E.g., Vulnerabilities in Medtronic's pacemakers and insulin pumps could be exploited by hackers to modify the system firmware, forcibly altering the device's operating voltage to 830V, thereby resulting in the death of patients undergoing cardiac resuscitation [42].

$f_{cp}^{sp} : Affected\ platform \rightarrow Target\ platform$: The state affected platform provide abnormal support to the target platform. E.g., code contributors inserted malicious code into the XZ Utils software, causing numerous Linux distributions that adopted the software to contain a backdoor, thereby exposing a large number of user hosts to vulnerabilities [43].

4.3.3 State-affect relations

F^{st} represents the state-affect relations generated by C_a on C_t through the intermediate components in C_r:

$$F^{st} = \{f_{data}^{st}, f_{code}^{st}\} \tag{14}$$

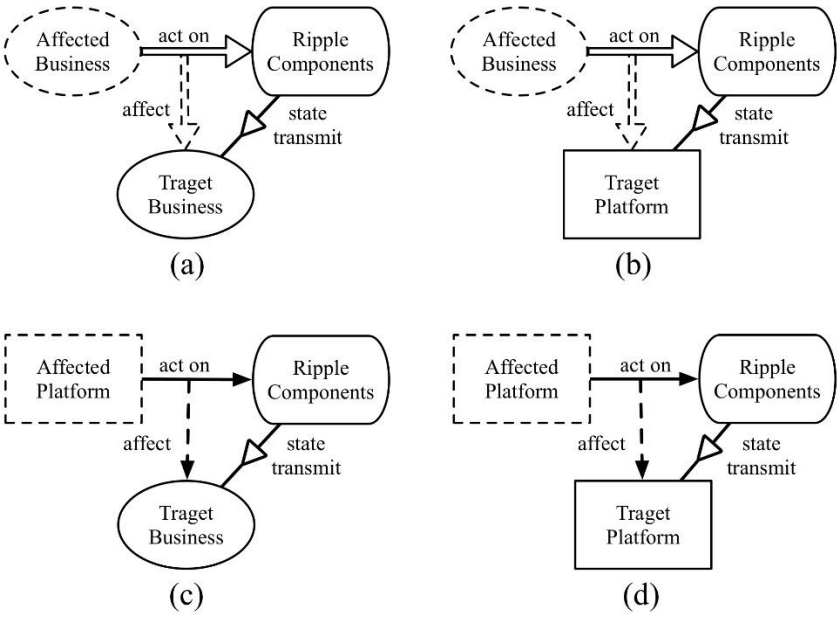


Figure 10. State-affect mapping relations. (a) and (b) represent state effect is caused by affected business through data streams, (c) and (d) represent that is caused by affected platform.

Here, f_{data}^{st} represents the state effect resulting from threat business data generated by the affected business. The mapping relation is expressed as:

$$f_{data}^{st}: Affected\ business \xrightarrow{C_r} C_t \tag{15}$$

Expending Eq.15, as shown in Figure 10(a) and 10(b), respectively:

$f_{data}^{st}: Affected\ business \xrightarrow{C_r} Target\ business$: The affected business transmits the abnormal state to the target business through combined relations. E.g., attackers hacked into the Belarusian Railway's internal network and closed all services, making it impossible for travelers to buy tickets and forcing the national railway system into manual control mode, which resulted in the slowing down of the trains and the paralysis of some railway junctions [46].

$f_{data}^{st}: Affected\ business \xrightarrow{C_r} Target\ platform$: The affected business transmits the abnormal state to the target platform through combined relations. E.g., in the Ukraine blackout, attackers launched the telephony DDOS offline attack against the electricity customer service center, causing the breakdown of customer service business. Inability to receive valid customer feedback that could be used to determine the area of the outage slowed down the power checking process, which delayed the restoration of the power facilities [34].

And f_{code}^{st} represents the state effect resulting from threat code data generated by the affected platform. The mapping relation is expressed as:

$$f_{code}^{st}: Affected\ platform \xrightarrow{C_r} C_t \tag{16}$$

Expending Eq.16, as shown in Figure 10(c) and 10(d), respectively:

f_{code}^{st} : $Affected\ platform \xrightarrow{C_r} Target\ business$: The affected platform transmits the abnormal state to the target business through combined relations. E.g., a ransomware attack on the data center of Colonial Pipeline, the largest refined products pipeline in the United States, forced the oil and gas pipeline system that delivers 45 per cent of the fuel supply to major cities on the East Coast offline, causing shortages of petrol, diesel and aviation fuel [36].

f_{code}^{st} : $Affected\ platform \xrightarrow{C_r} Target\ platform$: The affected platform transmits the abnormal state to the target business through combined relations. E.g., a cyberattack on European satellites led to a massive outage in communication, which prevented the monitoring and control system of the wind energy company Enercon from communicating remotely. This caused the remote control and monitoring ability for roughly 5800 wind turbines with a total of 11GW was lost, some turbines were damaged [44].

The above mapping relations describe the fundamental patterns of threat transmission in Model D, where Eq.9 and Eq.12 represent threat transmission through business streams, while Eq.10 and Eq.13 represent transmission code streams. Eq.15 and Eq.16 characterize spread via abnormal state transmission.

The RipA model uniformly abstracts threat transmission as the combination of attack actions and three mechanisms by which threats affect target, thereby capturing the domino effect of threat ripples. In practical scenarios, complex dependency structures may place a component within multiple dependency chains simultaneously, enabling threats to spread along multiple paths.

4.4. Model Representation

In this chapter, we construct the RipA model D = {C, H, F}. Where D is the set of objects in the model, including threat source，threat agent，affected Business，target Business, affected Platform, target Platform and ripple components.

H is the set of attack functions:

$$H = \{attack : threat\ source,\ threat\ agent \rightarrow C_a\} \tag{17}$$

F is the set of threat-affect functions:

$$\begin{aligned} F = \{&abnormal\ manage : C_a \rightarrow C_t, \\ &abnormal\ support : C_a \rightarrow C_t, \\ &state\ transmit : C_a \xrightarrow{C_r} C_t\} \end{aligned} \tag{18}$$

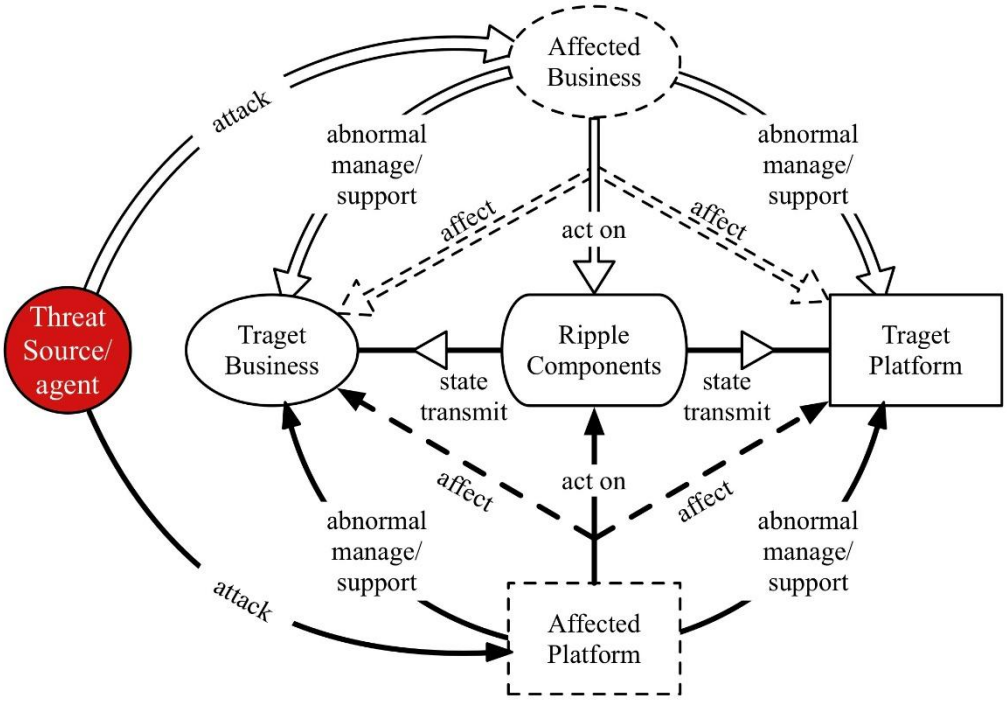


Figure 11 RipA model

The RipA model is able to characterize the attack logic of the adversary utilizing the dependencies among the platforms and the businesses to make the threat spread, as well as the domino effect in the attacks against informatization architectures. The RipA model is shown in Figure 11.

5 Mathematical Representation of Models

In this paper, mathematical representations are constructed for the architecture model and the RipA model, enabling algorithmic implementation and thereby supporting automated analysis and inference of threat ripples.

5.1 Objects Definition

The objects in architecture are defined as follows:

$$B = \{B_1, B_2, ..., B_t\} \tag{19}$$

where B is the set of business components. The supporting platform of business $B_i \in B$ is denoted as P_i. Thus, the set of supporting platforms is:

$$P = \{P_1, P_2, ..., P_t\} \tag{20}$$

The business that directly connected to business B_i is denoted by $B_j(i)$, and its corresponding supporting platform is denoted as $P_j(i)$. For any business or platform component $C_i \in B_i \bigcup P_i$, its state is

represented by S_{C_i}. The **secure state** of C_i is denoted as $S_{C_i}^N$, and the **abnormal state** is denoted as $S_{C_i}^F$. When subjected to an attack or threat ripple, the state notations remain unchanged if C_i is tolerant of intrusion and remains in secure state, or becomes abnormal without transmitting threats outward.

If C_i remains functions normally but becomes an intermediate component through which threats penetrate and continue to transmit, its state is denoted by $\tilde{S}_{C_i}^N$, representing the **penetration ripple state**.

If C_i becomes abnormal and simultaneously continues to transmit threats outward, the state is denoted by $\tilde{S}_{C_i}^F$, representing the **abnormal ripple state**, which is distinguished from abnormal states $S_{C_i}^F$ that do not transmit threats. In summary:

$$S_{C_i} \in \{S_{C_i}^N, S_{C_i}^F, \tilde{S}_{C_i}^N, \tilde{S}_{C_i}^F\}, \tilde{S}_{C_i} \in \{\tilde{S}_{C_i}^N, \tilde{S}_{C_i}^F\} \tag{21}$$

The continuous state vector of component C_i at time t is defined as:

$$State(S_{C_i}^t) = [\vec{H}_{C_i}^t, \vec{P}_{C_i}^t], \forall u_{C_i}^t \in \vec{H}_{C_i}^t, p_{C_i}^t \in \vec{P}_{C_i}^t, 0 \le u_{C_i}^t, p_{C_i}^t \le 1 \tag{22}$$

In Eq.22, the health status vector $\vec{H}_{C_i}^t$ represents the integrity of the component's capability. For each indicator $u_{C_i}^t$ in the vector (such as performance, operational error rate, etc.), 1 indicates complete health while 0 signifies total failure.

The threat severity vector $\vec{P}_{C_i}^t$ represents the degree to which the component's output information is contaminated. Each indicator $p_{C_i}^t$ follows a standard which ranges from 0 (pure output) to 1 (fully threat output). Let the health status and threat severity thresholds at time t be $\vec{\theta}_{C_i}^t$ and $\vec{\rho}_{C_i}^t$, respectively, then:

$$\begin{aligned} S_{C_i}^{N,t} &: \left[\!\left[\vec{H}_{C_i}^t > \vec{\theta}_{C_i}^t \right]\!\right] \wedge \left[\!\left[\vec{P}_{C_i}^t < \vec{\rho}_{C_i}^t \right]\!\right], \quad S_{C_i}^{F,t} : \left[\!\left[\vec{H}_{C_i}^t \le \vec{\theta}_{C_i}^t \right]\!\right] \wedge \left[\!\left[\vec{P}_{C_i}^t < \vec{\rho}_{C_i}^t \right]\!\right] \\ \tilde{S}_{C_i}^{N,t} &: \left[\!\left[\vec{H}_{C_i}^t > \vec{\theta}_{C_i}^t \right]\!\right] \wedge \left[\!\left[\vec{P}_{C_i}^t \ge \vec{\rho}_{C_i}^t \right]\!\right], \quad \tilde{S}_{C_i}^{F,t} : \left[\!\left[\vec{H}_{C_i}^t \le \vec{\theta}_{C_i}^t \right]\!\right] \wedge \left[\!\left[\vec{P}_{C_i}^t \ge \vec{\rho}_{C_i}^t \right]\!\right] \end{aligned} \tag{23}$$

In Eq.23, the double brackets indicate evaluation according to established rules: the inequality is deemed satisfied once specific items in the vector meet the conditions, without requiring all items to satisfy them. For example, even if some parameters $u_{C_i}^t, p_{C_i}^t$ may individually have low values, several parameters satisfying the conditions simultaneously may still indicate a change in the component state. In such cases, the inequality is also considered satisfied.

For the setting of parameters and thresholds, particularly when dealing with very large-scale heterogeneous

architectures, defenders may group components with similar functions, roles, and dependencies into the same category and assign a shared parameter and threshold template to each category. These templates may be initially estimated based on component characteristics, expert assessments, and attack exercise results rather than simulation experiments. Targeted adjustments can then be made for individual components based on their specific conditions. It should be noted that components located at the entry points of dependency chains or otherwise possessing the potential to trigger ripple effects require precise parameter calibration. For components whose ability to trigger ripple effects remains uncertain, conservative thresholds or threshold intervals may be adopted. In addition, these parameters and thresholds can be dynamically updated as new analytical data, event records, or inference results become available.

In modern information architectures, relationships among components are established through directed data transmission; therefore, the resulting dependency relations are dynamic.

We use $N^t(C_i, C_j(i))$ to describe the relation R formed among components at time t, and use $D^t_{C_i} \mid^R_{C_j(\mathrm{i})}$ to denote the secure data transmitted from component C_i to its directly connected component $C_j(\mathrm{i})$ [3]:

$$D^t_{C_i} \mid^R_{C_j(\mathrm{i})} \vDash N^t(C_i, C_j(\mathrm{i})) \tag{24}$$

In Eq.24, $R(C_i, C_j(i)) \in \{mg, sp, co\}$ denotes the type of relations among the components at time t, where mg, sp and co represent management, support, and cooperation relationship, respectively. The R in the formula serves as the relation label. When discussing attacks or threat ripples, $A^t_{C_i} \mid^R_{C_j(i)}$ is used to denote threat data.

5.2 Objects Definition

We design an attack function, a state function, an action function, and a ripple function to represent the fundamental logic of component interactions and threat transmission. Specifically, the attack function characterizes the triggering mechanism of abnormal states. The state function describes how a component outputs information, whereas the action function determines how external information changes the state of the component. The ripple function is used to express cross-component threat transmission.

5.2.1 Attack function

We use a structured vector to describe the attack behavior of adversary T:

[3] Specifically, for the purpose of abstract modeling, the case in which platform P_i runs code data to support its corresponding business B_i is regarded as a special form of transmitting code data outwards.

$$Attack(T) = [\vec{K}, \vec{X}_{C_i}], \quad where \quad k_r \in \vec{K}, x_m \in \vec{X}_{C_i} \tag{25}$$

In Eq.25， $\vec{K}$ denotes the capability vector of the adversary, and k_r denotes the r th attack capability indicator. This vector describes the intelligence resources, attack tools, participating personnel, funding, attack methods, and capabilities for penetrating complex networks possessed by the adversary. $\vec{X}_{C_i}$ denotes the vector of vulnerabilities & weaknesses that can be exploited in the target. Each $x_m \in \vec{X}_{C_i}$ characterizes a vulnerability or weak object by specifying its location, triggering conditions, type, and potential impacts.

A target component usually possesses certain defensive capabilities. This paper uses the structured vector $\overrightarrow{Def}^t(C_i)$ to describe its threat detection and blocking capabilities, emergency response speed, intrusion tolerance and recovery capability.

The intensity of an attack depends on the current state $State(S_{C_i}^t)$ of target component C_i, the attack behavior $Attack(T)$, and its current defensive capability $\overrightarrow{Def}^t(C_i)$. This paper uses vector $\vec{\Gamma}_{C_i}^t$ to represent the attack action intensity exerted by the adversary on component C_i at time t：

$$\vec{\Gamma}_{C_i}^t = Strength(\vec{K}, \vec{X}_{C_i}, \overrightarrow{Def}^t(C_i)) \tag{26}$$

In Eq.26, the function $Strength$ is calculated based on the attack capability $\vec{V}_K$, the vulnerability & weakness exploitability $\vec{V}_X$ and the current defensive capability of the target $\vec{V}_{Def}$, which are obtained by applying the corresponding weight vectors:

$$\begin{aligned} \vec{V}_K &= \vec{\omega}_K \odot \vec{K} \\ \vec{V}_X &= \vec{\omega}_X \odot \vec{X}_{C_i} \\ \vec{V}_{Def} &= \vec{\omega}_{Def} \odot \overrightarrow{Def}^t(C_i) \end{aligned} \tag{27}$$

In Eq.27, the elements of $\vec{\omega}_K$, $\vec{\omega}_X$ and $\vec{\omega}_{Def}$ represent the weights corresponding to the attack capability indicators $k_r \in \vec{K}$, the vulnerability & weakness indicators $x_m \in \vec{X}_{C_i}$ and the defensive capability indicators $def_l^t \in \overrightarrow{Def}^t(C_i)$, respectively. These weights can be determined using the analytic hierarchy process (AHP) based on attack mechanisms, vulnerability or weakness exploitation knowledge, expert assessments, the results of attack and defense exercises, or historical attack incident data. Since the numbers of indicators in the vectors may differ, $\Gamma_{C_i}^t$ is expressed as Eq.28:

$$\begin{aligned} &\vec{\Gamma}_{C_i}^t = \sigma(W_K \vec{V}_K + W_X \vec{V}_X - W_{Def} \vec{V}_{Def}) \\ &W_K \in \mathbb{R}^{n_\Gamma \times n_K}, W_X \in \mathbb{R}^{n_\Gamma \times n_X}, W_{Def} \in \mathbb{R}^{n_\Gamma \times n_{Def}} \end{aligned} \tag{28}$$

In Eq.28， σ is a normalization function used to constrain the attack action intensity to

the interval $[0,1]$. The attack action mapping matrices W_K, W_X and W_{Def} map the respective vectors into a unified attack action intensity space. Alternatively, the weight vectors described above may also be incorporated directly into their corresponding mapping matrices. The resulting state of target component C_i after an attack is expressed as $State(S_{C_i}^{t+\Delta t})$:

$$\vec{H}_{C_i}^{t+\Delta t} = \vec{H}_{C_i}^{t} - M_H \vec{\Gamma}_{C_i}^{t}, M_H \in \mathbb{R}^{n_H \times n_\Gamma}$$
$$\vec{P}_{C_i}^{t+\Delta t} = \vec{P}_{C_i}^{t} + M_P \Gamma_{C_i}^{t}, M_P \in \mathbb{R}^{n_H \times n_\Gamma} \quad (29)$$

Eq.29 determines the final state of the component. The impact mapping matrices M_H and M_P represent the attenuation relationships between different dimensions of attack action and the health status indicators, and the enhancement relationships between different dimensions of attack action and the threat severity indicators, respectively. This is because the dimensions of attack action do not necessarily correspond to those of the component state indicators. These matrices can be determined based on the operational or running mechanisms of the component in a specific scenario, expert experience, or historical attack incident data.

In applications, this paper defines the attack function h_{C_i} to provide a simplified representation of the attack process. From the defender's perspective, the attack result is reflected in the change in the state of the component at time t: the adversary exploits vulnerability or weakness x_m in component C_i, causing C_i to enter state $S_{C_i}^t$ at time t:

$$h_{C_i}(\cdot \mid x_m) \mapsto S_{C_i}^t \quad (30)$$

Certainly, the internal decomposition and calculation of the attack function can also be implemented according to specific application scenarios by using existing computational methods, such as weighting models, defense bypass probability functions, vulnerability exploitation functions, and damage estimation methods. Through reconnaissance and intelligence analysis, adversaries can determine, to some extent, how to design an attack based on x_m that causes the target to enter the abnormal ripple state $\tilde{S}_{C_i}^F$, thereby enabling the threat to continue transmitting.

5.2.2 State function

To accurately characterize data transmission attributes, we design the state function $I_{C_i}^t|_{C_j(i)}$ to describe the process by which component C_i transmits data to its directly connected component $C_j(i) \in [B_j(i), P_j(i)]$ at time t. Its input is the current state of component C_i, denoted as $State(S_{C_i}^t)$,

and its output is the data generated under that state. When component C_i in secure state $S_{C_i}^N$ outputs data to component $C_j(i)$ to establish relation R, we have:

$$I_{C_i}^t|_{C_j(i)}: S_{C_i}^N \rightarrow D_{C_i}^t|_{C_j(i)}^R \quad (31)$$

In addition, although data transmission is based on directly connected components, relations may be indirectly established across multiple components. That is, component C_i may provide management, support, or collaboration to the target component C_t indirectly through components $C_{i+1}, C_{i+2}, \cdots, C_{t-1}$. Accordingly, we introduce the subscript $<\mathrm{C}_t>=[\mathrm{C}_i, \mathrm{C}_{i+1}, \cdots, \mathrm{C}_t]$ to represent the path through which C_i establishes a relationship with C_t. Eq.31 then becomes:

$$I_{C_i}^t|_{<C_t>}: S_{C_i}^N \rightarrow D_{C_i}^t|_{<C_t>}^R \quad (32)$$

It should be noted that $<\mathrm{C}_t>$ merely records the path and does not alter any properties of the state function. The output data from component C_i must still first be transmitted to its directly connected component $\mathrm{C}_j(\mathrm{i})$ (here $\mathrm{C}_j(\mathrm{i})=\mathrm{C}_{i+1}$). The latter will forward or manufact the data according to its capability and task requirements, then transmits the resulting data to subsequent components. When $C_t=\mathrm{C}_j(\mathrm{i})$, we have $<\mathrm{C}_t>=\mathrm{C}_j(\mathrm{i})$, in this case Eq.31 and Eq.32 are identical.

If component C_i enters an abnormal ripple state after being subjected to a threat and subsequently outputs the threat data in full, Eq.32 thus becomes:

$$I_{C_i}^t|_{<C_t>}: \tilde{S}_{C_i}^F \rightarrow A_{C_i}^t|_{<C_t>}^R \quad (33)$$

If component C_i is in a penetration ripple state, means that although its own capability remains normal, it has become an intermediate node for threat transmission. Therefore, its output simultaneously contains both secure data and threat data:

$$I_{C_i}^t|_{<C_t>}: \tilde{S}_{C_i}^N \rightarrow D_{C_i}^t|_{<C_t>}^R \cdot \omega_D + A_{C_i}^t|_{<C_t>}^R \cdot \omega_A \quad (34)$$

where ω_D and ω_A represent the weights of secure data and threat data, respectively, $\omega_A, \omega_D \in [0,1], \omega_A + \omega_D = 1$. It should be noted that when a component is in an abnormal ripple state, this

does not mean that all of its capabilities have failed. Its output information may still contain safe data, as described in Eq.34. Besides, if component C_i enters a non-abnormal ripple state $S_{C_i}^F$, it does not transmit threats outward.

For the internal computation of the state function, the information output by component C_i to target component C_t through relation R at time t is jointly determined by its health status vector $\vec{H}_{C_i}^t$ and threat severity vector $\vec{P}_{C_i}^t$. Therefore, an **effectiveness vector** $\vec{\eta}_{C_i}^t|_{<C_t>}^R$ and a **contamination vector** $\vec{\mu}_{C_i}^t|_{<C_t>}^R$ are constructed to represent the characteristics of the output information:

$$\vec{\eta}_{C_i}^t|_{<C_t>}^R = W_{C_i}^H|_{<C_t>}^R \vec{H}_{C_i}^t, W_{C_i}^H|_{<C_t>}^R \in \mathbb{R}^{n_\eta \times n_H} \quad (35)$$

$$\vec{\mu}_{C_i}^t|_{<C_t>}^R = W_{C_i}^P|_{<C_t>}^R \vec{P}_{C_i}^t, W_{C_i}^P|_{<C_t>}^R \in \mathbb{R}^{n_\mu \times n_P} \quad (36)$$

$\vec{\eta}_{C_i}^t|_{<C_t>}^R$ in Eq.25 includes indicators such as completeness, field parseability, temporal consistency, and source credibility, representing the output capability of the component and the validity of its information. In Eq.26, $\vec{\mu}_{C_i}^t|_{<C_t>}^R$ includes indicators such as semantic deviation, malicious payload matching, the likelihood of field tampering, and the intensity of anomalous features, representing the degree of contamination of the output information. Together, these two vectors determine the information output mechanism.

Here, $W_{C_i}^H|_{<C_t>}^R \in \mathbb{R}^{n_\eta \times n_H}$ and $W_{C_i}^P|_{<C_t>}^R \in \mathbb{R}^{n_\mu \times n_P}$ are the mapping matrices corresponding to the effectiveness vector and the contamination vector, respectively. They contain the weights that reflect the effects of different indicators on the output information and ensure that $\vec{\eta}_{C_i}^t|_{<C_t>}^R$ and $\vec{\mu}_{C_i}^t|_{<C_t>}^R$ have consistent dimensions. The specific values of these weights are determined based on the dependency mechanisms between component C_i and target component C_t, as well as the interaction mechanisms between component C_i and its directly connected component $C_j(i)$ along in the chain, together with task requirements and historical experience.

For the output contents of normal information $D_{C_i}^t|_{<C_t>}^R$ and threat information $A_{C_i}^t|_{<C_t>}^R$, the state function $I_{C_i}^t|_{<C_t>}$ is essentially an indicator function associated with component C_i. It outputs the corresponding information when the specified condition is satisfied:

$$D_{C_i}^t|_{<C_t>}^R = I_{C_i}^t|_{<C_t>} [\![\vec{\eta}_{C_i}^t|_{<C_t>}^R \odot (\vec{e} - \vec{\mu}_{C_i}^t|_{<C_t>}^R) \geq \vec{\tau}_D^R]\!] \quad (37)$$

$$A_{C_i}^t|_{<C_t>}^R = I_{C_i}^t|_{<C_t>} [\![\vec{\eta}_{C_i}^t|_{<C_t>}^R \odot \vec{\mu}_{C_i}^t|_{<C_t>}^R \geq \vec{\tau}_A^R]\!] \quad (38)$$

Here, $\vec{e}$ denotes the unit vector, and $\vec{e} - \vec{\mu}_{C_i}^t|_{<C_t>}^R$ represents the degree to which the information remains uncontaminated. The condition calculation uses elementwise multiplication to express the coupling between output capability and contamination degree: the generation of normal information requires not only sufficient effective output capability,

but also a low degree of contamination in the output information. Similarly, the generation of threat information depends not only on the contamination degree, but also on whether the component, although abnormal, still retains the capability to transmit information outward. Therefore, the condition in Eq.37 characterizes the output intensity of normal information, whereas the condition in Eq.38 characterizes the output intensity of threat information. Besides, $\vec{\tau}_D^R$ and $\vec{\tau}_A^R$ denote the threshold vectors. The meaning of the double brackets has been explained above.

5.2.3 Action function

The action function E_{C_i} characterizes the internal processing of input data by component C_i. Its output is the new state formed by the component. This function can be coupled with the data transmission process characterized by the state function.

When component $C_j(i)$ receives and processes normal data $D_{C_i}^t|_{<C_t>}^R$ from component C_i at time t, its new state at time $t+\Delta t$ will become:

$$E_{C_j(i)}^{\Delta t}: D_{C_i}^t|_{<C_t>}^R \rightarrow S_{C_j(i)}^N \tag{39}$$

When the input is threat data $A_{C_i}^t|_{<C_t>}^R$, Eq.11 then becomes:

$$E_{C_j(i)}^{\Delta t}: A_{C_i}^t|_{<C_t>}^R \rightarrow S_{C_j(i)} \tag{40}$$

Eq.40 can be expanded into four cases, as shown in Table 2.

Table 2 Mathematical expressions of the component action function under threat

Mapping Relation	Explanation
$E_{C_j(i)}^{\Delta t}: A_{C_i}^t|_{<C_t>}^R \rightarrow S_{C_j(i)}^N$	The component tolerates the intrusion, and the threat no longer spreads.
$E_{C_j(i)}^{\Delta t}: A_{C_i}^t|_{<C_t>}^R \rightarrow S_{C_j(i)}^F$	The threat causes the component to enter an abnormal state, but the threat does not spread further.
$E_{C_j(i)}^{\Delta t}: A_{C_i}^t|_{<C_t>}^R \rightarrow \tilde{S}_{C_j(i)}^N$	The component enters a penetration ripple state, and the threat penetrates the component and continues to spread.
$E_{C_j(i)}^{\Delta t}: A_{C_i}^t|_{<C_t>}^R \rightarrow \tilde{S}_{C_j(i)}^F$	The component enters an abnormal ripple state, and the threat continues to spread.

For the internal computation of the action function, the mixed input information $O_{C_i}^t|_{C_j(i)}^R$ received by downstream component $C_j(i)$ from component C_i can be expressed as:

$$O_{C_i}^t|_{C_j(i)}^R = \omega_D \cdot D_{C_i}^t|_{<C_t>}^R + \omega_A \cdot A_{C_i}^t|_{<C_t>}^R \tag{41}$$

Since the downstream component cannot obtain the state of upstream component, including its health status and threat severity, or the output determination process of its state function, the calculation of state effect must be based on feature extraction from the input

information. Therefore, the function $\Phi_{C_j(i)}$ is constructed to extract the features that may change the state of the current component:

$$\vec{\varphi}^{t}_{C_i}\,|_{C_j(i)}, \vec{\delta}^{t}_{C_i}\,|_{C_j(i)} = \Phi_{C_j(i)}(O^{t}_{C_i}\,|^{R}_{C_j(i)}) \tag{42}$$

Here, the behavior vector $\vec{\varphi}^{t}_{C_i}$ indicates the effects produced by the tasks contained in the input information that need to be processed by $C_j(i)$. It includes risk indicators such as potential erroneous operations and parameter mismatches, and is used to reflect the behavioral effects that may be exerted on the current component. The contamination vector $\vec{\delta}^{t}_{C_i}$ indicates the degree of contamination of the input information. It includes indicators such as existing threats and anomalous data, and is used to reflect direct threats posed by external information. The dimensions of these two vectors are set to be the same.

These vectors will change the state of the downstream component, namely its health status vector and threat severity vector. Meanwhile, the component itself possesses certain defensive capabilities. When an anomaly is detected, these capabilities can prevent the execution of inappropriate tasks to a certain extent and attenuate external threats. Let $\vec{\tau}^{R}_{\varphi}$ and $\vec{\tau}^{R}_{\delta}$ denote the threshold vectors of the behavior vector $\vec{\varphi}^{t}_{C_i}$ and the contamination vector $\vec{\delta}^{t}_{C_i}$, respectively. Their values are jointly determined by the intrinsic properties and defensive capability $\overrightarrow{Def}^{t}(C_j(i))$ of component $C_j(i)$. The detected behavior anomaly indicator vector $\vec{m}^{\varphi}_{C_i}\,|_{C_j(i)}$ and contamination anomaly indicator vector $\vec{m}^{\delta}_{C_i}\,|_{C_j(i)}$ are respectively expressed as:

$$\vec{m}^{\varphi}_{C_i}\,|_{C_j(i)} = \hbar[\![\vec{\varphi}^{t}_{C_i}\,|^{R}_{C_j(i)} \geq \vec{\tau}^{R}_{\varphi}]\!] \tag{43}$$

$$\vec{m}^{\delta}_{C_i}\,|_{C_j(i)} = \hbar[\![\vec{\delta}^{t}_{C_i}\,|^{R}_{C_j(i)} \geq \vec{\tau}^{R}_{\delta}]\!] \tag{44}$$

Here, the function $\hbar$ denotes a threshold filtering function: when the corresponding indicator reaches the threshold, the indicator is retained; otherwise, it is set to 0. The defensive capability acts only on the indicators in the behavior vector and contamination vector that reach the thresholds, while indicators below the thresholds do not participate in defensive processing. Therefore, the suppression vector $\vec{d}^{H}_{C_i}\,|_{C_j(i)}$ of the component defensive capability against risky behaviors and the attenuation vector $\vec{d}^{P}_{C_i}\,|_{C_j(i)}$ for the contamination degree are defined as:

$$\vec{d}^{H}_{C_i}\,|_{C_j(i)} = \vec{m}^{\varphi}_{C_i}\,|_{C_j(i)} \odot (M^{H}_{C_i}\,|_{C_j(i)}\,\overrightarrow{Def}^{t}(C_j(i))), M^{H}_{C_i}\,|_{C_j(i)} \in \mathbb{R}^{n_{\varphi} \times n_{Def}} \tag{45}$$

$$\vec{d}^{P}_{C_i}\,|_{C_j(i)} = \vec{m}^{\delta}_{C_i}\,|_{C_j(i)} \odot (M^{P}_{C_i}\,|_{C_j(i)}\,\overrightarrow{Def}^{t}(C_j(i))), M^{P}_{C_i}\,|_{C_j(i)} \in \mathbb{R}^{n_{\delta} \times n_{Def}} \tag{46}$$

Here, the mapping matrices $M_{C_i}^{H}|_{C_j(i)}$ and $M_{C_i}^{P}|_{C_j(i)}$ represent the mappings between the defensive capability vector $\overrightarrow{Def}^{t}(C_j(i))$ and the behavior anomaly indicators and contamination anomaly indicators, respectively: permission verification, blocking of abnormal commands, etc., can ensure the safety of task execution to a certain extent, whereas intrusion detection, anomalous traffic identification, etc., can attenuate the contamination effects of threat information. Therefore, the behavior vector $\vec{\varphi}'_{C_i}|_{C_j(i)}$ and $\vec{\delta}'_{C_i}|_{C_j(i)}$ contamination vector that ultimately act on the component state are respectively expressed as:

$$\vec{\varphi}'_{C_i}|_{C_j(i)} = \vec{\varphi}^{t}_{C_i}|_{C_j(i)} \odot (\vec{e} - \vec{d}^{H}_{C_i}|_{C_j(i)}) \tag{47}$$

$$\vec{\delta}'_{C_i}|_{C_j(i)} = \vec{\delta}^{t}_{C_i}|_{C_j(i)} \odot (\vec{e} - \vec{d}^{P}_{C_i}|_{C_j(i)}) \tag{48}$$

The effects of the input information on the state of the current component, namely its health status and threat severity, can be expressed as:

$$\vec{H}^{t+\Delta t}_{C_j(i)} = \vec{H}^{t}_{C_j(i)} - M^{\varphi}_{H}\vec{\varphi}'_{C_i}|_{C_j(i)} - M^{\delta}_{H}\vec{\delta}'_{C_i}|_{C_j(i)} \tag{49}$$

$$\vec{P}^{t+\Delta t}_{C_j(i)} = \vec{P}^{t}_{C_j(i)} + M^{\varphi}_{P}\vec{\varphi}'_{C_i}|_{C_j(i)} + M^{\delta}_{P}\vec{\delta}'_{C_i}|_{C_j(i)} \tag{50}$$

Here, M^{φ}_{H} and M^{δ}_{H} denote the effect mapping matrices of the behavior vector and contamination vector on the health status vector, while M^{φ}_{P} and M^{δ}_{P} denote their effect mapping matrices on the threat severity vector, respectively. This is because risky behaviors and contamination effects reduce the health status of the component while increasing its threat severity.

5.2.4 Ripple function

We construct the ripple function $F_{C_i}|^{R}_{C_j(i)}$ by composing the state function and the action function to represent the fundamental ripple effect logic among components after an attack occurs. Its input is the component's ripple state $\tilde{S}_{C_i}$, and its output is the resulting state of the directly connected component. In essence, it is a composite mapping of the state function and the action function with respect to threat data $A^{t}_{C_i}|^{R}_{<C_t>}$: component C_i in ripple state $\tilde{S}_{C_i}$ outputs threat data and affects the state of component $C_j(i)$. This process can be expressed based on Eq.33, Eq.34 and Eq.40 as:

$$\begin{cases} I^{t}_{C_i}|_{<C_t>} : \tilde{S}_{C_i} \rightarrow A^{t}_{C_i}|^{R}_{<C_t>} \\ E^{\Delta t}_{C_j(i)} : A^{t}_{C_i}|^{R}_{<C_t>} \rightarrow S_{C_j(i)} \end{cases} \tag{51}$$

By composing the mappings in Eq.13, we have the ripple function $F_{C_i}|^{R}_{C_j(i)}$ of component C_i in state

$\tilde{S}_{C_i}$:

$$F_{C_i}|^{R}_{<C_t>}: \tilde{S}_{C_i} \rightarrow S_{C_j(i)} \tag{52}$$

Expanding Eq. (14), there are four mapping relationships, as shown in Table 3.

Table 3 Mathematical expressions of the ripple function in the Threat transmission Process

Mapping Relation	Explanation	Equation
$F_{C_i}\|^{R}_{<C_t>}: \tilde{S}_{C_i} \rightarrow S^{N}_{C_j(i)}$	Component $C_j(i)$ is tolerant of intrusion from component C_i, in this case, $<C_t>=C_j(i)$.	$F_{C_i}\|^{R}_{<C_t>}(\tilde{S}_{C_i}) = S^{N}_{C_j(i)}$
$F_{C_i}\|^{R}_{<C_t>}: \tilde{S}_{C_i} \rightarrow S^{F}_{C_j(i)}$	The abnormal state of component C_i results in an anomalous state in component $C_j(i)$, but the threat will no longer spread, $<C_t>=C_j(i)$.	$F_{C_i}\|^{R}_{<C_t>}(\tilde{S}_{C_i}) = S^{F}_{C_j(i)}$
$F_{C_i}\|^{R}_{<C_t>}: \tilde{S}_{C_i} \rightarrow \tilde{S}^{N}_{C_j(i)}$	The abnormal state of component C_i does not render component $C_j(i)$ abnormal, but the threat will penetrate $C_j(i)$ and continue to spread.	$F_{C_i}\|^{R}_{<C_t>}(\tilde{S}_{C_i}) = \tilde{S}^{N}_{C_j(i)}$
$F_{C_i}\|^{R}_{<C_t>}: \tilde{S}_{C_i} \rightarrow \tilde{S}^{F}_{C_j(i)}$	The abnormal state of component C_i results in an anomalous state in component $C_j(i)$, and the threat will continue to spread, $<C_t>=C_j(i)$.	$F_{C_i}\|^{R}_{<C_t>}(\tilde{S}_{C_i}) = \tilde{S}^{F}_{C_j(i)}$

5.3 Ripple Effect

This section builds upon the previously constructed attack function and ripple function to clarify the reachability conditions under which a threat ripples from an upstream component to a target component, given predefined component dependency relations, thereby characterizing the ripple effect of attack behaviors on architecture state evolution. When the target component C_t cannot be attacked directly, and its secure state depends on secure management, support, or cooperation provided by its directly connected component C_{t-1}, if an adversary can exploit the weakness x in component C_{t-1} to drive it into an abnormal ripple state $\tilde{S}^{F}_{C_{t-1}}$, then the adversary may attempt to transmit threats to component C_t as follows:

$$S^{F}_{C_t} = F_{C_{t-1}}|^{R}_{C_t} \cdot h_{C_{t-1}}(\cdot \mid x) \tag{53}$$

If there exists a weakness x satisfying Eq.53, then once an adversary exploits x to render component

C_{t-1} abnormal, thereby transmitting threats to component C_t and causing it to become abnormal.

We extend Eq. 15 as follows. Consider a r a dependency sequence $C_1, C_2, ..., C_{t-1}, C_t$, in which the normal state of any platform or business component C_i depends on the normal state of its prior component $C_{i-1}, i \in [2, t]$. When the adversary is unable to directly attack the target component C_t, the ripple functions of its preceding dependent components under ripple states, namely $F_{C_{t-1}}|_{C_t}^R, F_{C_{t-2}}|_{C_{t-1}}^R, \ldots$ can be recursively substituted into the solution form of Eq.53, until a ripple function $F_{C_i}|_{C_{i+1}}^R$ is identified for which a solution exists for $\tilde{S}_{C_i}^F$. That is,

$$\begin{aligned} S_{C_t}^F &= F_{C_{t-1}}|_{C_t}^R(\tilde{S}_{C_{t-1}}) \\ \tilde{S}_{C_{t-1}} &= F_{C_{t-2}}|_{C_{t-1}}^R(\tilde{S}_{C_{t-2}}) \\ \tilde{S}_{C_{t-2}} &= F_{C_{t-3}}|_{C_{t-2}}^R(\tilde{S}_{C_{t-3}}) \\ &\cdots \\ \tilde{S}_{C_{i+1}} &= F_{C_i}|_{C_{i+1}}^R(\tilde{S}_{C_i}^F) \end{aligned} \tag{54}$$

Combining the above equations in Eq.54 and substituting $h_{C_i}(\cdot \mid x) \mapsto \tilde{S}_{C_i}^F$, we obtain:

$$S_{C_t}^F = F_{C_{t-1}}|_{C_t}^R \cdot F_{C_{t-2}}|_{C_{t-1}}^R \cdots F_{C_i}|_{C_{i+1}}^R \cdot h_{C_i}(\cdot \mid x) \tag{55}$$

When there exists a set of components in the dependency sequence that satisfies Eq.16, that is, when an adversary T exploits weakness x to launch an attack on an upstream component C_i within the dependency sequence, the threat will naturally transmit from the abnormal ripple state $\tilde{S}_{C_i}^F$ through $\tilde{S}_{C_{i+1}}, \tilde{S}_{C_{i+2}}, ..., \tilde{S}_{C_{t-1}}$, ultimately rendering the target component C_t becomes abnormal and enters the state $S_{C_t}^F$.

It should be noted that the above formulation is intended solely to illustrate the transmission mechanism and realization conditions of threat ripple within dependency relationships, and does not involve the specific computation algorithms or implementation procedures for solving attack paths. The state effect logic following $\tilde{S}_{C_i}^F$ is determined by the architecture's operational mode rather than by the adversary. However, the adversary can obtain partial information through intelligence gathering, active penetration, and similar means.

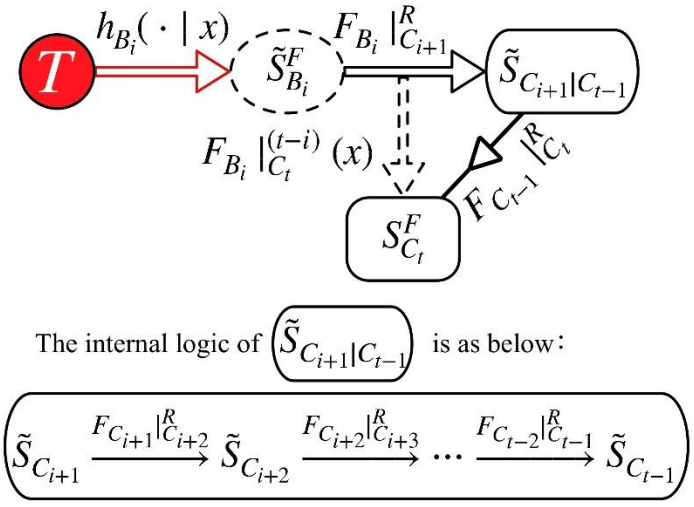


Figure 12 Mathematical representation of ripple effect from the adversary's perspective

In ripple effects, we are usually focus on the formation mechanism of abnormal state transmission and its eventual consequences. For Eq.55, we construct the function $F_{C_i}|^{(m)}_{C_t}(x)$ to aggregate the final ripple effect on the informatization architecture caused by the threat from the adversary, as shown in Figure 12, so as to avoid repeatedly expanding and redundant elaboration to the threat transmission process. Here, the superscript (m) denotes the **ripple depth**, that is, the length of the dependency chain that the threat transmits outward from the attacked component can affect. The subscript C_t represents the component that is ultimately affected at ripple depth m.

For the case where an adversary exploits weakness x to attack component C_i, ultimately causing the target component $C_{i+m} = C_t$ to enter an abnormal state, the expression is given as:

$$F_{C_i}|^{(t-1)}_{C_t}(x) = S^F_{C_t} \tag{56}$$

In particular, if component C_i does not enter a ripple state after being attacked, the threat transmission terminates, and the ripple depth is 0 at this time.

5.4 Mathematical Formulation of the Model

This section presents the mathematical expressions of the functions defined previously and explains how they are mapped onto the informatization architecture model in section 3 and the RipA model in section 4

5.4.1 Informatization System Model

We use a combination of state functions and effect functions to characterize the operating and running modes of information business B_I, non-information business B_{NI}, and platform P (including CPS) within the informatization architecture model, the manner in which they are carried by the functions defined above, as well as the mapping of the interconnections among these functions at the object level. The mathematical

expressions of the informatization architecture model are shown in Figure 13[4].

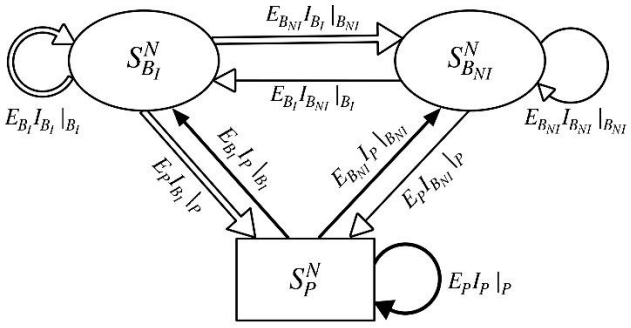


Figure 13 Mathematical expressions of the informatization architecture model

In the process of describing and modeling the operational mechanism of the architecture, state function and action function instances can be directly constructed based on the function reference in Table 4 and in combination with the actual scenario, thereby mapping the aforementioned model onto real-world informatization architecture objects. This enables the characterization of how external inputs to each object affect its current state, as well as the resulting changes in outputs under the new state. When discussing threat ripple, the ripple function shown in Table 3 can be applied, as appropriate, to express which other objects' states may be affected by an object currently in an abnormal state. This allows operators to characterize the architecture's dependency relations and information interaction patterns, thereby identifying potential threat transmission paths.

Table 4 Function reference table for the informatization architecture model

object[5]	input	Action function	State function	Output	Ripple function
C_{cur}	$D^t_{C_{in}}\|^R_{C_{cur}}$	$E^{\Delta t}_{B_I}$	$I^{t+\Delta t}_{C_i}\|_{<C_i>}$	$D^{t+\Delta t}_{C_{cur}}\|^R_{C_{out}}$	$F_{C_{cur}}\|^R_{C_{out}}(\tilde{S}_{C_{cur}}) = S_{C_{out}}$

5.4.2 RipA Model

We use T to denote the threat source or threat agent, B_A and P_A to denote the affected origin business and affected origin platform, respectively. B_T and P_T denote the target business and target platform in the process of threat ripple. The mathematical expressions of the RipA model are shown in Figure 14.

In the model, we employ the functions defined above to characterize the effects and transmission mechanisms of threats within the architecture. For **attack behaviors**, we use h_{B_A} and h_{P_A} to represent the

[4] The workflow generated by non-information business may contain information data.

[5] C_{cur} denotes the object currently under analysis, while C_{in} and C_{out} denote the input object and output object that exchange data with C_{cur} during the function mapping process, respectively.

attack mappings from T to B_A and P_A respectively, namely h_{data} and h_{code}, which will trigger the abnormal ripple states of the origin components.

After the attack occurs, the threats transmit toward the targets leverage the dependencies within the architecture. For the effects caused by abnormal management, we use $F_{B_A}|_{C_T}^{mg}$ and $F_{P_A}|_{C_T}^{mg}$ to represent the **management-affect** mappings f_{data}^{mg} and f_{code}^{mg} from B_A and P_A to the target C_T, respectively. For the effects caused by abnormal support, we use $F_{B_A}|_{C_T}^{sp}$ and $F_{P_A}|_{C_T}^{sp}$ to represent the **support-affect** mappings f_{data}^{sp} and f_{code}^{sp}. These constitute the foundational threat effect patterns.

Additionally, for **state-affect** relations composed of multiple management-affect relations and support-affect relations, we use $F_{B_A}|_{C_T}^{(m)}$ and $F_{P_A}|_{C_T}^{(m)}$ to denote the state-affect mappings f_{data}^{st} and f_{code}^{st} initiated by B_A and P_A, so as to describe the domino effects generated by the spread of abnormal states within the architecture.

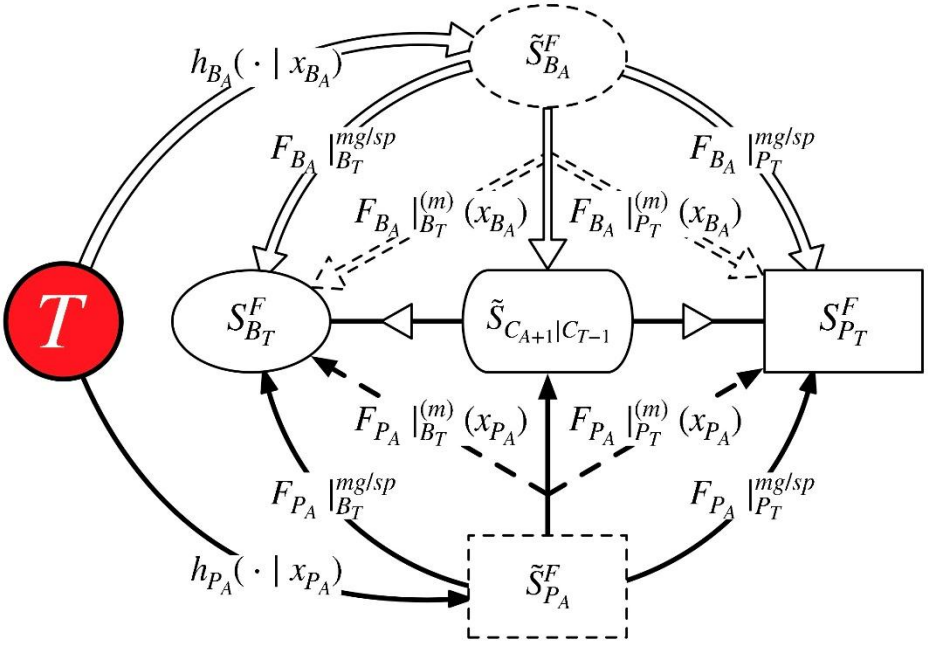


Figure 14 Mathematical expressions of the RipA model

When describing a threat event, operators can, based on the function reference provided in Table 3, replace the input of the attacked component with threat data or threat code, or adjust its action function or state function according to the component's abnormal state, thereby computing the effect of the threat on that component. During subsequent threat effect computation, the threat output of the upstream component's state function can be taken as the input to the downstream component's action function, so as to characterize the state evolution process and ripple effects triggered by the threat within the architecture. At the same time, in combination with the ripple functions in the RipA model, it is possible to formally express how abnormal states transmit by

leveraging the operational modes and dependency relations within the architecture, thereby providing a computable analytical basis for attack assessment and threat interruption.

This paper establishes a unified mathematical description framework for the informatization architecture model and the RipA model, formalizing threat transmission logic into composable and derivable functional relationships. The framework is able to support quantitative analysis and computation of the threat ripple process, laying the foundation for algorithmic research. In practice, the specific contents of various attack functions and ripple functions (including the effect functions and state functions that constitute them) can be defined according to the application scenario, thereby endowing the model with high extensibility and generalization capability.

5.4.3 Flexible Application of the Model

We present a case study of a real incident. On July 19, 2024, during routine operations and maintenance, CrowdStrike delivered a Rapid Response Content configuration update to Falcon Sensor on Windows endpoints, namely Channel File 291, corresponding to the C00000291*.sys file. Due to logical defects in the template definition and array bounds validation, some Falcon Sensors on Windows endpoints accessed invalid memory regions after loading the update, which triggered system crashes and caused blue screens and repeated reboots. This incident was not a conventional cyber intrusion attack, but a global information system outage caused by a defective security software content update. Microsoft estimated that approximately 8.5 million Windows devices were affected: due to CrowdStrike was widely deployed in critical business environments such as aviation, finance, healthcare, cloud services, and public services, a localized technical failure rapidly developed into business disruptions across multiple industries [32]. The knowledge representation of this incident is shown in Figure 15. This incident demonstrates that, in an informatization architecture, even a single configuration defect may become a source of cascading risk. Abnormal effects can rapidly transmit through dependencies and cause industry level consequences far beyond the initial point of failure.

When constructing the model, this paper does not attempt to reproduce all loading and invocation details of the internal modules of Windows endpoints during the update process. Instead, based on the management dependencies of several important businesses on the affected endpoints and the support dependencies of those endpoints on the content distribution server, the informatization architecture model is used to characterize the operational mechanism shown in Figure 16(a). Based on the incident knowledge, the RipA model is then used to represent the state effect process among components shown in Figure 16(b), while Table 5 presents the corresponding model functions.

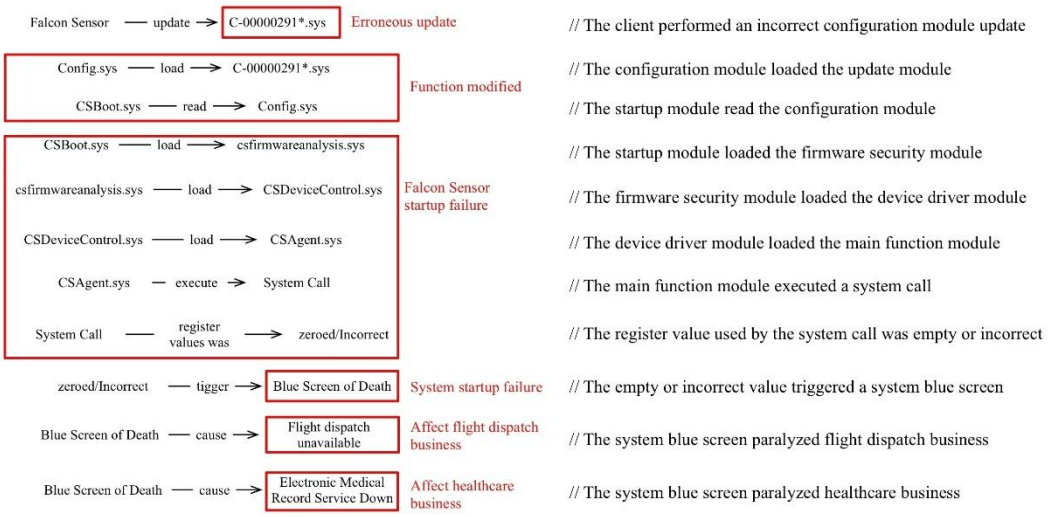
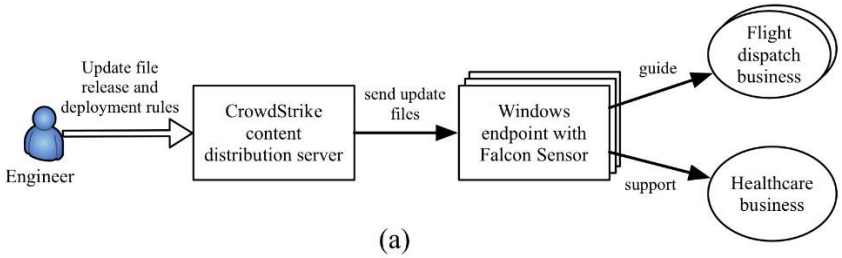
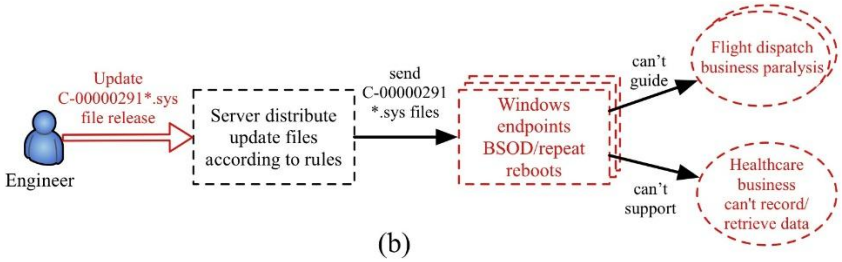


Figure 15 Knowledge representation of ripple impacts in the CrowdStrike blue screen incident.

It should be noted that although section 5 provides complete mathematical expression rules, researchers may further instantiate the state function, action function, and ripple function according to specific scenarios, thereby calculating component state changes, threat transmission conditions, and ripple depth. However, in many real threat incidents, the research focus is often on rapidly identifying the origin of an abnormal state, its transmission path, the affected components, and possible blocking points, rather than precisely quantifying every stage. Therefore, when parameters are insufficient, incident details are not fully disclosed, or the analysis mainly concerns mechanism interpretation and formal representation, researchers may directly use the RipA model and its mathematical framework to qualitatively model the incident. When the required parameters are available and more detailed inference is needed, corresponding computational analysis can then be conducted. This usage, which supports computation, simplification, and instantiation as needed, demonstrates the flexibility of the proposed modeling method.



(a)



(b)

Figure 16 CrowdStrike blue screen incident. (a) The operational mechanism represented by the informatization architecture model. (b) The ripple impact represented by the RipA model.

Table 5 Comparison table of the threat model functions for CrowdStrike blue screen incident

object	input	Action function	State function	Output	Ripple function
CrowdStrike content distribution server	File upload	Loads the C00000291*.sys update file and distribution rules	Distribute the C00000291*.sys update to Falcon Sensor	Abnormal sys update package	Abnormal update package → Windows endpoint failures
Windows endpoint with Falcon Sensor installed	Abnormal sys update package	Incorrectly modifies configuration data and crashes after an erroneous system call	Blue screen of death and repeated reboots	Output interruption	Windows endpoint failures → downstream business abnormal
Aviation dispatch business	Input interruption	Loses operational guidance, business paralysis	—	—	—
Healthcare business	Input interruption	Fails to record or retrieve information, business paralysis	—	—	—

6 Characterize Typical Threat Scenario

For informatization architectures in real scenarios, vendors usually keep their operational information confidential, including specific business capabilities, operating parameters of platform devices, component dependencies, and current state data. Public threat reports on attack incidents also tend to disclose only the attack process and resulting impacts, while omitting the underlying data required for reproducible experiments. Under such circumstances, directly adopting assumed parameters and simulation models may make experiments easy to construct, but would introduce subjective settings detached from actual business constraints and provide limited support for practical security defense. Therefore, this paper does not conduct parameterized simulations or reproduction experiments based on real power system data. Instead, it characterizes the operational mechanisms of a typical power production scenario based on existing materials and related studies. Furthermore, this paper combines these mechanisms with publicly presented executable attack patterns to conduct scenario reasoning on how threats transmit along business dependencies and generate ripple effects.

Papers [51-55] describe typical power production scenarios. Paper [56] proposes an attack method against a power production architecture, focusing on how an adversary can use false data injection to induce modifications to network parameters. However, it does not explain from the perspective of business dependencies why localized data tampering can cause consecutive abnormalities across multiple business processes, including state estimation, error detection, data

recording, parameter identification, control correction, and production operations. Therefore, without a threat model oriented toward dependencies among businesses, defenders may observe the attack input and final consequences but still find it difficult to determine how the threat transmits within the architecture, which critical components it passes through, how deeply the ripple effect extends, and where blocking measures should be implemented. This paper uses this attack as a representative case to demonstrate how defenders can apply the proposed informatization architecture model in practice to construct the interaction logic and dependencies of a given scenario, and how the RipA model can be used to analyze the ripple effects caused by the attack.

6.1 Attack Scenario

This paper take the power production scenario as an attack scenario, describe the state monitoring and estimation processing business process of power production, and construct its architecture model in the monitor mode and alarm mode, respectively.

6.1.1 State Monitor and Estimation Process

The operation parameters of power production may have errors due to untimely updating, errors in updating, malicious modifications, etc., resulting in deviation of the power production status from the expectation. The state monitoring and estimation processing business workflow is the process of utilizing collected power data to estimate the current production state, and correcting the current production state with historical data once a state abnormality has been detected. During this workflow, the architecture will be in monitor mode or alarm mode.

In the monitor mode, the architecture monitors the power production state in real time through the network monitor stations and collects the power production data, which is then sent to the state estimator to compute the estimates of the operating parameters (referred to as state estimates) that correspond to the current production state. If the state estimates meet the standards, it is considered that the production state is normal and the data is also entered into the database [51]. While any of the state estimates is found to be non-standard, the architecture considers the power production state to be abnormal and will enter the alarm mode. At this point, the control center will regenerate the state estimates based on the historical data from the database and then use them to adjust the power production operation parameters. For the power architecture, a correct and stable production state is very important, which, if disrupted, can have very serious consequences such as large-scale power blackouts [55].

In order to detect errors in power operation parameters timely and correctly, it is necessary to ensure the accuracy of data measurements at the network monitor station. Only by minimizing measurement errors can the errors in state estimates be more accurately recovered, thus allowing operators to modify operation parameters in time. Therefore, when the control center evaluates the power production state, it also evaluates the measurement accuracy. Once any deviation from the accuracy is detected, the measurement accuracy of the monitor station is corrected by modify the network parameters to avoid influencing the erroneous data judgment in the state estimation.

6.1.2 Monitor Mode

The architecture model of the state monitoring and estimation processing business of power production in

the monitor mode is shown in Figure 17. The model shows the workflow of data collection, state estimation and error judgment. For concise description, the feedback streams of data and code transmissions are hidden in the model. During power production, the **network monitor station** is responsible for collecting data at each connection cable and generating the actual measurement vector $\vec{z}$. The measurement algorithm can be expressed as:

$$\vec{z} = h(\vec{x}, \vec{p}) + \vec{e} \tag{57}$$

Where h is the nonlinear state estimation function for the two measurements "state of the production system" and "network parameters with errors". $x = [V_1 \cdots V_n, \theta_1 \cdots \theta_n] \in \mathbb{R}^{2n-1}$ is the state vector of power production including voltage magnitude V and phase angle θ, n is the number of the buses. $\vec{p}$ is the vector containing network parameter errors. $\vec{e}$ is the vector of measurement errors, is usually considered to be a random Gaussian variable with zero mean and covariance matrix G [54].

After obtaining the actual measurement vector $\vec{z}$, the **state estimator** computes the state estimation vector $\vec{x}_c$ by minimizing the following objective function J using the traditional weighted least squares(WLS) estimation approach. At this point, to ensure that the calculation is as accurate as possible, the network parameter errors are normally assumed to be zero by state estimator:

$$J = (\vec{z} - h(\vec{x}_c))^T \cdot G^{-1} \cdot (\vec{z} - h(\vec{x}_c)) \tag{58}$$

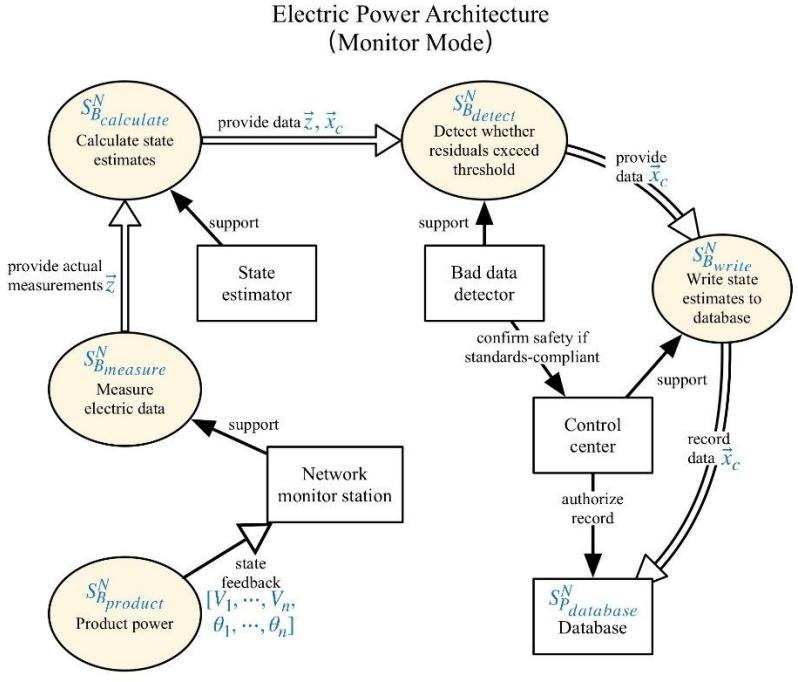


Figure 17 Architecture model of the power production in the monitor mode

Next, the state estimator sends the actual measurement vector $\vec{z}$ and the calculated state estimation vector $\vec{x}_c$ to the **bad data detector** to detect whether the measurement residual exceeds the security threshold. The largest normalized residual test (LNRT)-based bad data detection approach takes the difference between the

actual measurement $\vec{z}$ and the $h(x_c)$ generated from $\vec{x}_c$ as the measurement residual vector $\vec{r}$, which will be normalized under the confidence level of 99.7% subsequently. The processed results will be compared to the threshold λ (usually $\lambda = 3$) to judge whether the value of the i th measurement z_i meets the standard [57], as shown in Eq.59. If satisfied, the **control center** will write the state estimation vector $\vec{x}_c$ into the **database**:

$$r_i^N = \frac{|r_i|}{\sqrt{G_{ii}S_{ii}}} \le \lambda, \forall i \in M_l, l \in A \tag{59}$$

Where M_l is the set of measurements at the branch l, A is the set of branches in the power production system. G_{ii} is the i th diagonal element of the covariance matrix G, S_{ii} is the i th diagonal element of the residual sensitivity matrix $S = I - H \cdot (H^T \cdot G^{-1} \cdot H) \cdot H^T \cdot G$, where H is the Jacobian matrix derived from h. The comparison table of the architecture model functions in the monitor mode is shown in Table 6.

Table 6 Comparison table of the architecture model functions in the monitor mode

Object	Input	Action function	State function	Output
$B_{measure}$	$\vec{x}$	Solve for $\vec{z} = h(\vec{x}, \vec{p}) + \vec{e}$	Provide $\vec{z}$ to $B_{calculate}$	$\vec{z}$
$B_{calculate}$	$\vec{z}$	Minimize $J = (\vec{z} - h(\vec{x}_c))^T \cdot G^{-1} \cdot (\vec{z} - h(\vec{x}_c))$ to solve for $\vec{x}_c$	Provide $\vec{z}, \vec{x}_c$ to B_{detect}	$\vec{z}, \vec{x}_c$
B_{detect}	$\vec{z}, \vec{x}_c$	Solve for $\vec{r} = \vec{z} - h(\vec{x}_c)$ and substitute into $r_i^N = \frac{\lvert r_i \rvert}{\sqrt{G_{ii}S_{ii}}}$	Find $r_i^N \le \lambda$, then provide $\vec{x}_c$ to B_{write}	$\vec{x}_c$
B_{write}	$\vec{x}_c$	Receive $\vec{x}_c$	Write $\vec{x}_c$ to $P_{database}$	$\vec{x}_c$
$P_{database}$	$\vec{x}_c$	Record $\vec{x}_c$ as $\vec{x}$	—	—

6.1.3 Alarm Mode

If the bad data detector detects that the value in the measurement residual vector exceeds the security

threshold (i.e.: $r_i^N > \lambda$), it alarms the control center for error parameter processing. This indicates that there is an abnormality in the power production state or/and an error in the network monitor station, at which point the architecture will enter the alarm mode, as shown in Figure 18.

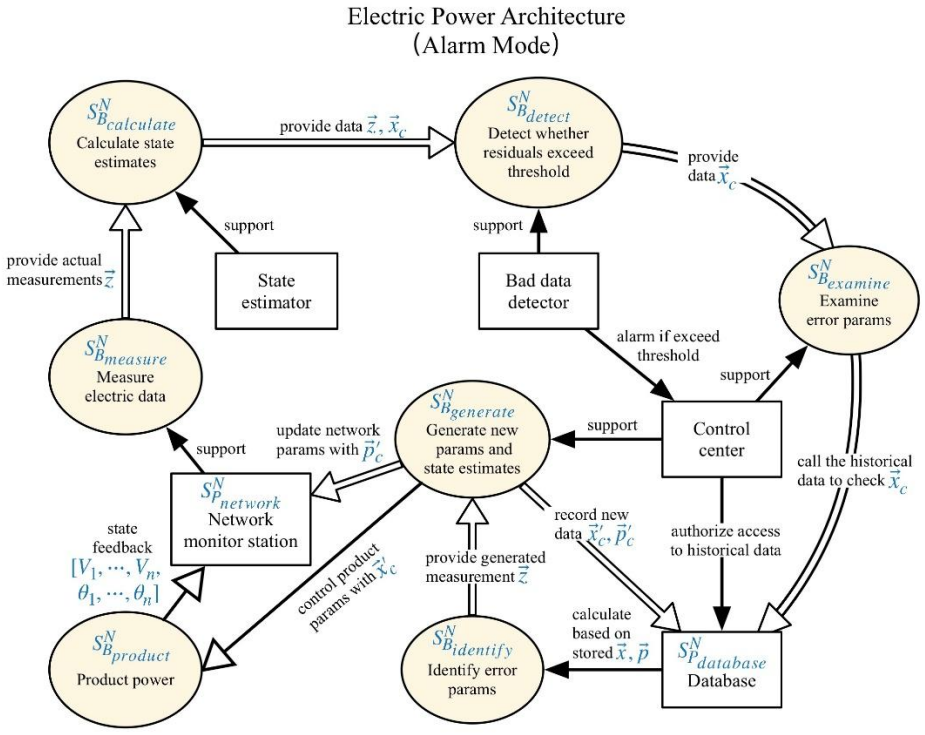


Figure 18. Architecture model of the power production in the alarm mode

Upon receiving an alarm, the control center immediately indicates the examination business to call the recent state estimation vector $\vec{x}$ and the previous network parameters $\vec{p}$ from the authorized database to help with the error parameter identification. Erroneous network parameters, although do not directly affect the measurements, can distort the calculation process of state estimation and lead to biased results. Therefore, it is important to ensure the accuracy of the network parameters before correcting the state of power production. In terms of calculating, since the effect of the erroneous network parameter on the state estimation is the same as the effect brought by a set of correlated errors in measurements values with the erroneous branches (namely the power flows through the branch and the power injections at the end nodes), thus, a simple transformation of the basic measurement algorithm can check for monitor stations with erroneous network parameters:

$$\begin{aligned} \vec{z}_e &= h(\vec{x}, \vec{p}_e) + \vec{e} \\ &= h(\vec{x}, \vec{p}) + [h(\vec{x}, \vec{p}_e) - h(\vec{x}, \vec{p})] + \vec{e} \end{aligned} \tag{60}$$

Where $\vec{z}_e$ refers to the vector of erroneous measurements involved, $\vec{p}_e$ is the bad network parameter[6]. The term in square brackets in the above equation is equivalent to the additional measurement error, which can be equivalently expressed linear as:

[6] When the network parameter error $\vec{p}$ exceeds the allowable accuracy, it is considered to be bad.

$$h(\vec{x},\vec{p}_e)-h(\vec{x},\vec{p})=\left[\frac{\partial h}{\partial p}\right]\Delta\vec{p} \tag{61}$$

Where is $\Delta\vec{p}=\vec{p}_e-\vec{p}$ the network parameter error, it leads to inaccuracies in the values of the corresponding measurements. Similar to the state estimation approach, the Network Parameters Augmented State Estimation (NPASE) approach for correction can be formulated as a WLS problem [53], which aims at minimizing the following equation:

$$J=(\vec{z}-h(\vec{x},\vec{p}))^T\cdot G^{-1}\cdot(\vec{z}-h(\vec{x},\vec{p})) \tag{62}$$

That is, the measurement vector $\vec{z}$ which generated from the previously stored data, will be used to recalculate the state estimation vector $\vec{x}_c$ that contains the bus voltage magnitude and the bus phase angle, and the network estimation vector $\vec{p}_c$, so as to obtain the new correct state parameters and network parameters. The first-order optimality conditions for both are, respectively:

$$H^T(\vec{x}'_c,\vec{p}'_c)\cdot G^{-1}\cdot[\vec{z}-h(\vec{x}'_c,\vec{p}'_c)]=0 \tag{63}$$

$$H_p^T(\vec{x}'_c,\vec{p}'_c)\cdot G^{-1}\cdot[\vec{z}-h(\vec{x}'_c,\vec{p}'_c)]=0 \tag{64}$$

where H and H_p are the Jacobian matrices of h with respect to $\vec{x}'_c$ and $\vec{p}'_c$, respectively. These two equations can be distinguished according to whether $\vec{x}'_c$ and $\vec{p}'_c$ are determined simultaneously or sequentially. Finally, the control center uses $\vec{p}'_c$ to correct the network parameters of the monitor stations in to restore their accuracy, and uses the corresponding operation parameters in $\vec{x}'_c$ to control the power production facilities, while recording them in into the database. Up to this point, the error parameter processing workflow is completed. The comparison table of the architecture model functions in the alarm mode is shown in Table 7.

The control center then returns to the monitor mode and observes whether the bad data detector alarms after correcting error parameters. If everything is normal, the adjustment is considered successful.

Table 7 Comparison table of the architecture model functions in the alarm mode

Object	Input	Action function	State function	Output
$B_{measure}$	$\vec{x}$	Solve for $\vec{z}=h(\vec{x},\vec{p}_e)+\vec{e}$	Provide $\vec{z}$ to $B_{calculate}$	$\vec{z}$
$B_{calculate}$	$\vec{x},\vec{p}$	Minimize $J=(\vec{z}-h(\vec{x}_c))^T\cdot G^{-1}\cdot(\vec{z}-h(\vec{x}_c))$ to solve for $\vec{x}_c$	Provide $\vec{z},\vec{x}_c$ to B_{detect}	$\vec{z},\vec{x}_c$
B_{detect}	$\vec{z},\vec{x}_c$	Solve for $\vec{r}=\vec{z}-h(\vec{x}_c)$ and	Find $r_i^N>\lambda$,	$\vec{x}_c$

Object	Input	Action function	State function	Output
		substitute into $r_i^N = \frac{\lvert r_i \rvert}{\sqrt{G_{ii}S_{ii}}}$	then provide abnormal $\vec{x}_c$ to $B_{examine}$ Feedback the results to P_{detect}	State signal
P_{detect}	State signal	Trigger alarm process	Alarm to $P_{control}$	Alarm signal
$P_{control}$	Alarm signal	Trigger examine process	Indicate $B_{examine}$ to examine Authorize access to historical data to $P_{database}$ Indicate $B_{generate}$ to generate new params	Examination signal Authorization Param signal
$B_{examine}$	$\vec{x}_c$, Examination signal	Receive $\vec{x}_c$ and perform checks	Retrieve historical data for $\vec{x}_c$	$\vec{x}_c$
$P_{database}$	$\vec{x}_c$, authorization	Locate the values in the historical $\vec{x}$ corresponding to the error values in $\vec{x}_c$,together with the previous network params $\vec{p}$	Provide $\vec{x}$ and $\vec{p}$ to support $B_{identify}$ in identifying the error params.	$\vec{x}, \vec{p}$
$B_{identify}$	$\vec{x}, \vec{p}$	Based on $\vec{z} = h(\vec{x}, \vec{p})$ and $\vec{z} = h(\vec{x}, \vec{p}) + [h(\vec{x}, \vec{p}_e) - h(\vec{x}, \vec{p})] + \vec{e}, h(\vec{x}, \vec{p}_e) - h(\vec{x}, \vec{p}) = \left[\frac{\partial h}{\partial p}\right]\Delta\vec{p}$ to determine location m corresponding to abnormal $\Delta\vec{p}$	Mark position m and provide $\vec{z}_m$ to $B_{generate}$	$\vec{z}_m$
$B_{generate}$	$\vec{z}_m$, Param signal	Minimize $J = (\vec{z}_m - h(\vec{x}_c, \vec{p}_c))^T \cdot G^{-1} \cdot (\vec{z}_m - h(\vec{x}_c, \vec{p}_c))$ to solve for $\vec{x}_c, \vec{p}_c$, update data at position m	Write $\vec{x}_c, \vec{p}_c$ to $P_{database}$ Configure $P_{network}$ with $\vec{p}_c$ Correct $B_{product}$ with $\vec{x}_c$	$\vec{x}_c, \vec{p}_c$ $\vec{p}_c$ $\vec{x}_c$
$P_{database}$	$\vec{x}_c, \vec{p}_c$	Record new $\vec{x}_c, \vec{p}_c$	—	—
$P_{network}$	$\vec{p}_c$	Update the network params of the	—	—

Object	Input	Action function	State function	Output
		monitor stations with $\vec{p}_c$		
$B_{product}$	$\vec{x}_c$	Power production using data with $\vec{x}_c$	—	—

6.2 Threat Modeling

The goal of the adversary is to make the power production abnormal. If the control center can be successfully attacked, or the database used by the control center to configure the power system can be invaded and tampered with, the state of power production can be maliciously modified. However, due to the power architecture's layered management, sub-domain protection and other defense in-depth mechanisms, the conditions for the adversary to launch an attack are limited:

1. the adversary cannot access the well-secured control center or invade the associated database;

2. the adversary can only get incomplete information about the electric power architecture, gain a small part of true measurements, and cannot obtain a complete picture of the branch admittance and the production state;

3. the adversary has the ability to modify only a small part of the measurements in the power grid.

After gathering intelligence, the adversary finds that the operation parameter data recorded into the database is not obtained directly, but is calculated utilizing measurements, which can be affected by the network parameters. Thus, the adversary shifts the attack perspective to methods that can affect the network parameters. The whole attack is divided into three stages as follows:

Stage 1: The adversary exploits the bad data detection and parameter recording mechanisms of the power architecture to trick the control center write incorrect operation parameter data into the database by injecting false data into vulnerable network monitor stations[7].

Stage 2: The adversary injects false data which can cause the detector to alarm, mislead the control center to believe that there is a problem with the power production and use the database to perform a false correction to the network parameters and power production parameters.

Stage 3: The adversary injects new false data to make the control center believe that the corrections it has performed are correct and valid, thus maintaining the wrong state of power production.

The specific attack actions and threat ripple process are as follows:

[7] It's actually an attack against the business supported by network monitor stations through data stream.

6.2.1 Stage 1 of the attack

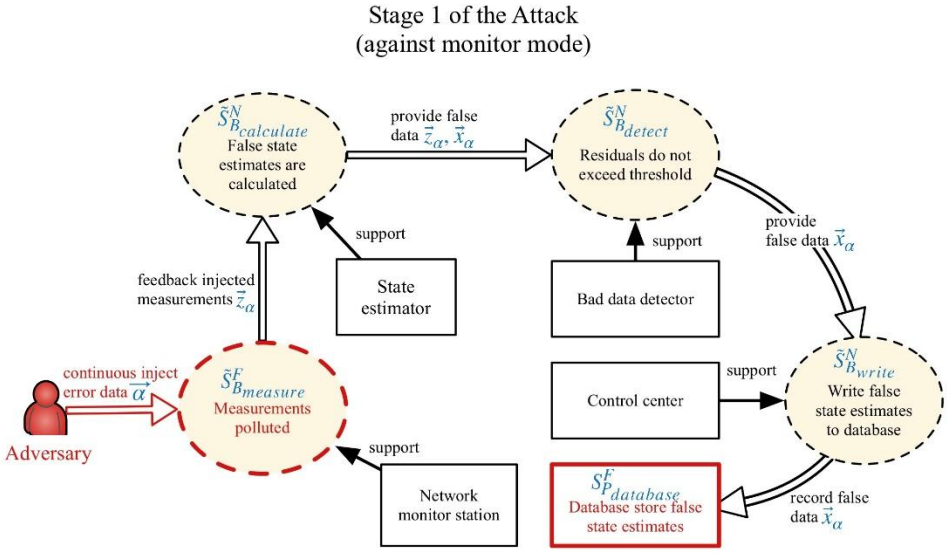


Figure 19 Threat model for the stage 1 of the power architecture attack

The threat model for the stage 1 of the attack against the power architecture is shown in Figure 19. The adversary continuously injects false data $\vec{\alpha}$ into vulnerable monitor stations, resulting in a gradual and slow increase of the actual measurements in these stations. At this point, the tampered actual measurement vector is noted as $\vec{z}_\alpha$:

$$\begin{aligned} \vec{z}_\alpha &= \vec{z} + \vec{\alpha} \\ &= h(\vec{x}, \vec{p}_e) + \vec{\alpha} + \vec{e} \\ &= h(\vec{x}, \vec{p}_e + \Delta\vec{p}_a) + \vec{e} \end{aligned} \tag{65}$$

This results in a subsequent alteration to the state estimates calculated by the state estimator. During this process, the adversary precisely adjusts the injected data so that the calculated residual $\vec{r}_\alpha$ always remains below the threshold of the bad data detector:

$$\vec{r}_\alpha = \vec{z}_\alpha - h(\vec{x}_\alpha) \tag{66}$$

$$r_i^N(\alpha) = \frac{|r_i(\alpha)|}{\sqrt{G_{ii}S_{ii}}} \le \lambda \tag{67}$$

This will cause the control center to incorrectly believe that the state estimation vector $\vec{x}_\alpha$ is valid and therefore write it into the database. The comparison table of the threat model functions is shown in Table 8. Due to space limitations in the tables and equations, when expressing ripple functions, the subscripts of component objects use the first three letters of their names. And the whole process can be expressed by the ripple functions as:

$$\begin{aligned} &h_{B_{mea}}(\cdot \mid \vec{\alpha}) \mapsto \tilde{S}^{F,t_1}_{B_{mea}} \\ &F_{B_{wri}}\big|^{mg}_{P_{dat}} \cdot F_{B_{det}}\big|^{sp}_{B_{wri}} \cdot F_{B_{cal}}\big|^{co}_{B_{det}} \cdot F_{B_{mea}}\big|^{sp}_{<B_{det}>}(\tilde{S}^{F,t_1}_{B_{mea}}) = S^{F,t_2}_{P_{dat}} \end{aligned} \tag{68}$$

Table 8 Comparison table of the threat model functions for the stage 1 of the attack

Object	Input	Action function	State function	Output	Ripple function
B_{measure}	$\vec{\alpha}$	Solve for $\vec{z}_\alpha = h(\vec{x}, \vec{p}_e) + \vec{\alpha} + \vec{e}$	Provide error $\vec{z}_\alpha$ to $B_{\text{calculate}}$	$\vec{z}_\alpha$	$h_{B_{mea}}(\cdot \mid \vec{\alpha}) \mapsto \tilde{S}^{F,t_1}_{B_{mea}}$ $F_{B_{mea}} \mid^{sp}_{<B_{\text{det}}>} (\tilde{S}^{F,t_1}_{B_{mea}}) = \tilde{S}^{N}_{B_{cal}}$
$B_{\text{calculate}}$	$\vec{z}_\alpha$	Minimize $J = (\vec{z}_\alpha - h(\vec{x}_\alpha))^T \cdot G^{-1} \cdot (\vec{z}_\alpha - h(\vec{x}_\alpha))$ to solve for $\vec{x}_\alpha$	Provide error $\vec{z}_\alpha, \vec{x}_\alpha$ to B_{detect}	$\vec{z}_\alpha, \vec{x}_\alpha$	$F_{B_{cal}} \mid^{co}_{B_{\text{det}}} (\tilde{S}^{N}_{B_{cal}}) = \tilde{S}^{N}_{B_{\text{det}}}$
B_{detect}	$\vec{z}_\alpha, \vec{x}_\alpha$	Solve for $\vec{r}_\alpha = \vec{z}_\alpha - h(\vec{x}_\alpha)$ and substitute into $r_i^N(\alpha) = \frac{\lvert r_i(\alpha) \rvert}{\sqrt{G_{ii} S_{ii}}}$	Find $r_i^N(\alpha) \le \lambda$ then provide error $\vec{x}_\alpha$ to B_{write}	$\vec{x}_\alpha$	$F_{B_{\text{det}}} \mid^{sp}_{B_{wri}} (\tilde{S}^{N}_{B_{\text{det}}}) = \tilde{S}^{N}_{B_{wri}}$
B_{write}	$\vec{x}_\alpha$	Receive $\vec{x}_\alpha$	Write error $\vec{x}_\alpha$ to $P_{database}$	$\vec{x}_\alpha$	$F_{B_{wri}} \mid^{mg}_{P_{dat}} (\tilde{S}^{N}_{B_{wri}}) = S^{F,t_2}_{P_{dat}}$
$P_{database}$	$\vec{x}_\alpha$	Record error $\vec{x}_\alpha$	—	—	—

The mathematical representation of the ripple effect in the stage 1 of the attack is shown in Figure 20. The adversary injects false data $\vec{\alpha}$ into the vulnerable monitor stations, causing the affected business B_{measure} to ripple the threat to platform $P_{database}$ at depth 4 in the dependency chain, thus achieving the goal of writing the error data to the database, i.e.

$$F^{t_1}_{B_{mea}} \mid^{(4)}_{P_{\text{dat}}} (\vec{\alpha}) = S^{F,t_2}_{P_{\text{data}}} \tag{69}$$

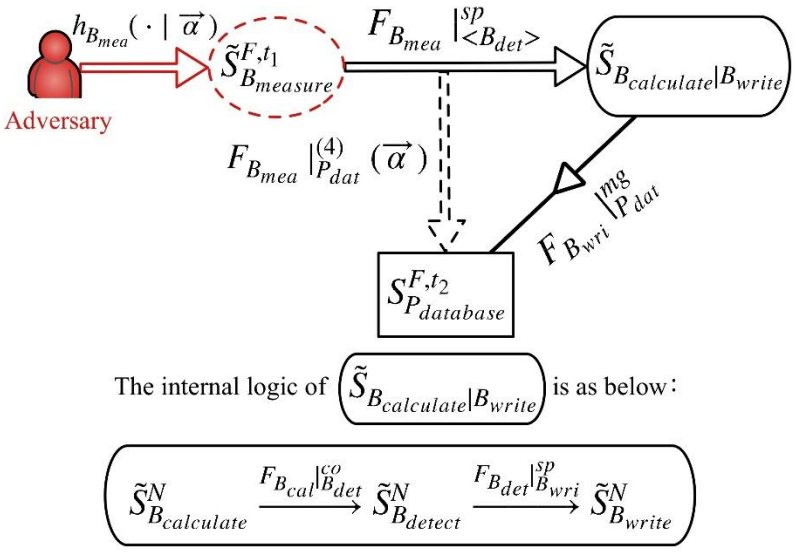


Figure 20 Mathematical expression of the ripple effect in the stage 1 of the attack

6.2.2 Stage 2 of the attack

The threat model for the stage 2 of the attack against the power architecture is shown in Figure 21. The adversary continues to inject the false data $\vec{\beta}$ that can cause abrupt changes in the actual measurements. This will make the state estimator to calculate state estimates $\vec{x}_{\beta}$ with large biases. When transmitted to the bad data detector, the corresponding measurement residuals will exceed the security threshold, causing the detector to trigger an alarm to the control center. Since the database has been polluted, the recent status measurement $\vec{x}_{\alpha}$ called up by the control center from it is incorrect, so is the measurement $\vec{z}_u$ generated with it. This affects the accuracy check of the network parameters, causing the control center to mistakenly believe that the network parameters of some monitor stations need to be corrected, as well as the state of the power production. Along with these incorrect state judgments, new network parameters $\vec{p}_u$ and state estimates $\vec{x}_u$ are calculated. Similarly, they are incorrect.

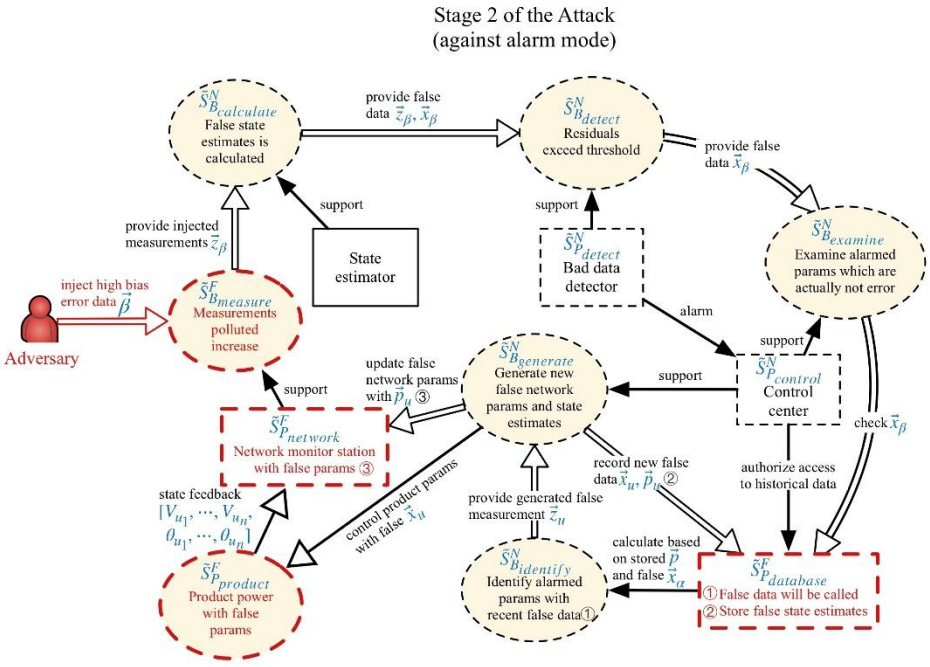


Figure 21 Threat model for the stage 2 of the power architecture attack

This causes the control center to modify the original normal network parameters and use the incorrect state parameters to control power production, while the database records the incorrect data. After the adjustments have been made, the control center continues to focus on the subsequent feedback state estimates to determine whether the adjustments are effective. The comparison table of the threat model functions is shown in Table 9, and the whole process can be expressed by the ripple functions as:

$$
\begin{aligned}
&h_{B_{mea}}(\cdot\,|\,\vec{\beta}) \mapsto \tilde{S}^{F,t_3}_{B_{mea}} \\
&F_{B_{\det}}\big|^{sp}_{B_{exa}}\big|^{mg}_{P_{\det}} \cdot F_{B_{cal}}\big|^{co}_{B_{\det}} \cdot F_{B_{mea}}\big|^{sp}_{<B_{\det}>}(\tilde{S}^{F,t_3}_{B_{mea}}) = \tilde{S}^{N,t_4}_{B_{exa}}, \tilde{S}^{N,t_4}_{P_{\det}} \\
&F_{P_{con}}\big|^{sp}_{B_{exa}}\big|^{mg}_{P_{dat}}\big|^{sp}_{B_{gen}} \cdot F_{P_{\det}}\big|^{sp}_{P_{con}}(\tilde{S}^{N,t_4}_{P_{\det}}) = \tilde{S}^{N,t_5}_{P_{dat}}, \tilde{S}^{N,t_5}_{B_{exa}}, \tilde{S}^{N,t_5}_{B_{gen}} \\
&F_{B_{exa}}\big|^{mg}_{P_{dat}}(\tilde{S}^{N,t_5}_{B_{exa}}) = \tilde{S}^{F,t_6}_{P_{dat}} \\
&F_{B_{ide}}\big|^{sp}_{B_{gen}} \cdot F_{P_{dat}}\big|^{sp}_{B_{ide}}(\tilde{S}^{F,t_6}_{P_{dat}}) = \tilde{S}^{\mathrm{N},t_7}_{B_{gen}} \qquad (70) \\
&F_{B_{gen}}\big|^{mg}_{P_{dat}}\big|^{mg}_{P_{net}}\big|^{mg}_{B_{pro}}(\tilde{S}^{\mathrm{N},t_7}_{B_{gen}}) = \begin{cases} E_{P_{dat}} \cdot I_{B_{gen}}\big|^{mg}_{P_{dat}}(\tilde{S}^{\mathrm{N},t_7}_{B_{gen}}) = S^{F,t_8}_{P_{dat}} \\ E_{P_{net}} \cdot I_{B_{gen}}\big|^{mg}_{P_{net}}(\tilde{S}^{\mathrm{N},t_7}_{B_{gen}}) = \tilde{S}^{F,t_8}_{P_{net}} \\ E_{B_{pro}} \cdot I_{B_{gen}}\big|^{mg}_{B_{pro}}(\tilde{S}^{\mathrm{N},t_7}_{B_{gen}}) = \tilde{S}^{F,t_8}_{B_{pro}} \end{cases}
\end{aligned}
$$

Table 9 Comparison table of the threat model functions for the stage 2 of the attack

Object	Input	Action function	State function	Output	Ripple function
B_{measure}	$\vec{\beta}$	Solve for $\vec{z}_\beta = h(\vec{x}, \vec{p}_e) + \vec{\beta} + \vec{e}$	Provide error $\vec{z}_\beta$ to $B_{\text{calculate}}$	$\vec{z}_\beta$	$h_{B_{mea}}(\cdot\,\|\,\vec{\beta}) \mapsto \tilde{S}^{F,t_3}_{B_{mea}}$ $F_{B_{mea}}\|^{sp}_{<B_{\det}>}(\tilde{S}^{F,t_3}_{B_{mea}}) = \tilde{S}^{N}_{B_{cal}}$
$B_{\text{calculate}}$	$\vec{z}_\beta$	Minimize	Provide error	$\vec{z}_\beta, \vec{x}_\beta$	$F_{B_{cal}}\|^{co}_{B_{\det}}(\tilde{S}^{N}_{B_{cal}}) = \tilde{S}^{N}_{B_{\det}}$

		$J=(\vec{z}_\beta - h(\vec{x}_\beta))^T \cdot G^{-1} \cdot(\vec{z}_\beta - h(\vec{x}_\beta))$ to solve for $\vec{x}_\beta$	$\vec{z}_\beta, \vec{x}_\beta$ to B_{detect}		
B_{detect}	$\vec{z}_\beta, \vec{x}_\beta$	Solve for $\vec{r}_\beta = \vec{z}_\beta - h(\vec{x}_\beta)$ and substitute into $r_i^N(\beta) = \frac{\lvert r_i(\beta)\rvert}{\sqrt{G_{ii}S_{ii}}}$	Find $r_i^N(\beta) > \lambda$,then provide abnormal $\vec{x}_\beta$ to $B_{examine}$	$\vec{x}_\beta$	$F_{B_{\text{det}}}\lvert^{sp}_{B_{exa}}(\tilde{S}^N_{B_{\text{det}}}) = \tilde{S}^{N,t_4}_{B_{exa}}$
			Feedback the results to P_{detect}	State signal	$F_{B_{\text{det}}}\lvert^{mg}_{P_{\text{det}}}(\tilde{S}^N_{B_{\text{det}}}) = \tilde{S}^{N,t_4}_{P_{\text{det}}}$
P_{detect}	State signal	Trigger an alarm process that should not have been executed.	Report a nonexistent alarm to $P_{control}$	Alarm signal	$F_{P_{\text{det}}}\lvert^{sp}_{P_{con}}(\tilde{S}^{N,t_4}_{P_{\text{det}}}) = \tilde{S}^N_{P_{con}}$
$P_{control}$	Alarm signal	Trigger examine process	Indicate $B_{examine}$ to examine	Examination signal	$F_{P_{con}}\lvert^{sp}_{B_{exa}}(\tilde{S}^N_{P_{con}}) = \tilde{S}^{N,t_5}_{B_{exa}}$
			Authorize access to historical data to $P_{database}$	Authorization	$F_{P_{con}}\lvert^{mg}_{P_{dat}}(\tilde{S}^N_{P_{con}}) = \tilde{S}^{N,t_5}_{P_{\text{dat}}}$
			Indicate $B_{generate}$ to generate new params for $\vec{x}_\beta$	Param signal	$F_{P_{con}}\lvert^{sp}_{B_{gen}}(\tilde{S}^N_{P_{con}}) = \tilde{S}^{N,t_5}_{B_{gen}}$
$B_{examine}$	$\vec{x}_\beta$, Examination signal	Receive $\vec{x}_\beta$ and perform checks	Retrieve historical data for $\vec{x}_\beta$	$\vec{x}_\beta$	$F_{B_{exa}}\lvert^{mg}_{P_{dat}}(\tilde{S}^{N,t_5}_{B_{exa}}) = \tilde{S}^{F,t_6}_{P_{dat}}$ [8]
$P_{database}$	Authorization	Locate the values in the historical $\vec{x}_\alpha$ corresponding to the error values in $\vec{x}_\beta$ (which is also erroneous), together with the previous network params $\vec{p}$	Provide error $\vec{x}_\alpha$ and $\vec{p}$ to support $B_{identify}$ in identifying the error params in $\vec{x}_\beta$	$\vec{x}_\alpha, \vec{p}$	$F_{P_{dat}}\lvert^{sp}_{B_{ide}}(\tilde{S}^{F,t_6}_{P_{dat}}) = \tilde{S}^{\text{N}}_{B_{ide}}$
$B_{identify}$	$\vec{x}_\alpha, \vec{p}$	Based on $\vec{z}_u = h(\vec{x}_\alpha, \vec{p})$ and $\vec{z}_u = h(\vec{x}_\alpha, \vec{p}) + [h(\vec{x}_\alpha, \vec{p}_e) - h(\vec{x}_\alpha, \vec{p})] + \vec{e}, h(\vec{x}_\alpha, \vec{p}_e) - h(\vec{x}_\alpha, \vec{p}) = \left[\frac{\partial h}{\partial p}\right]\Delta\vec{p}$	Mark position u and provide $\vec{z}_u$ to $B_{generate}$	$\vec{z}_u$	$F_{B_{ide}}\lvert^{sp}_{B_{gen}}(\tilde{S}^{\text{N}}_{B_{ide}}) = \tilde{S}^{\text{N},t_7}_{B_{gen}}$

[8] The database state change from $\tilde{S}^{N,t_5}_{P_{\text{dat}}}$ to $\tilde{S}^{F,t_6}_{P_{dat}}$ starts when the historical error data is called up.

		to determine location u corresponding to abnormal $\Delta\vec{p}$			
$B_{generate}$	$\vec{z}_u$, Param signal	Minimize $J=(\vec{z}_u-h(\vec{x}_u,\vec{p}_u))^T\cdot G^{-1}\cdot(\vec{z}_u-h(\vec{x}_u,\vec{p}_u))$ and get error $\vec{x}_u,\vec{p}_u$, then update data at position u	Write error $\vec{x}_u,\vec{p}_u$ to $P_{database}$	$\vec{x}_u,\vec{p}_u$	$E_{P_{dat}}\cdot I_{B_{gen}}\|^{mg}_{P_{dat}}(\tilde{S}^{\mathrm{N},t_7}_{B_{gen}})=S^{F,t_8}_{P_{dat}}$
			Configure $P_{network}$ using error $\vec{p}_u$	$\vec{p}_u$	$E_{P_{net}}\cdot I_{B_{gen}}\|^{mg}_{P_{net}}(\tilde{S}^{\mathrm{N},t_7}_{B_{gen}})=\tilde{S}^{F,t_8}_{P_{net}}$
			Configure $B_{product}$ using error $\vec{x}_u$	$\vec{x}_u$	$E_{B_{pro}}\cdot I_{B_{gen}}\|^{mg}_{B_{pro}}(\tilde{S}^{\mathrm{N},t_7}_{B_{gen}})=\tilde{S}^{F,t_8}_{B_{pro}}$
$P_{database}$	$\vec{x}_u,\vec{p}_u$	Record new x_u,p_u which are error	—	—	—
$B_{measure}$	$\vec{p}_u$	Update the network params of the monitor stations with error $\vec{p}_u$	—	—	—
$B_{product}$	$\vec{x}_u$	Use error data $\vec{x}_u$ to conduct abnormal power production	—	—	—

The mathematical representation of the ripple effect in the stage 2 of the attack is shown in Figure 22. The adversary injects false data $\vec{\beta}$ into the vulnerable monitor stations, causing the affected business B_{measure} to ripple the threat to platform $P_{network}$, $B_{product}$ and $P_{database}$ at depth 9 in the dependency chain, thus achieving the goal of inducing the power architecture to incorrectly modify the network parameters and power production state and to enter the generated error data $\vec{x}_u,\vec{p}_u$ into the database[9], i.e.:

$$F^{t_3}_{B_{mea}}\|^{(9)}_{P_{dat}}\|^{(9)}_{P_{net}}\|^{(9)}_{B_{pro}}(\vec{\beta})=S^{F,t_8}_{P_{dat}},\tilde{S}^{F,t_8}_{P_{net}},\tilde{S}^{F,t_8}_{B_{pro}} \tag{71}$$

[9] Threat ripple may form a ripple ring.

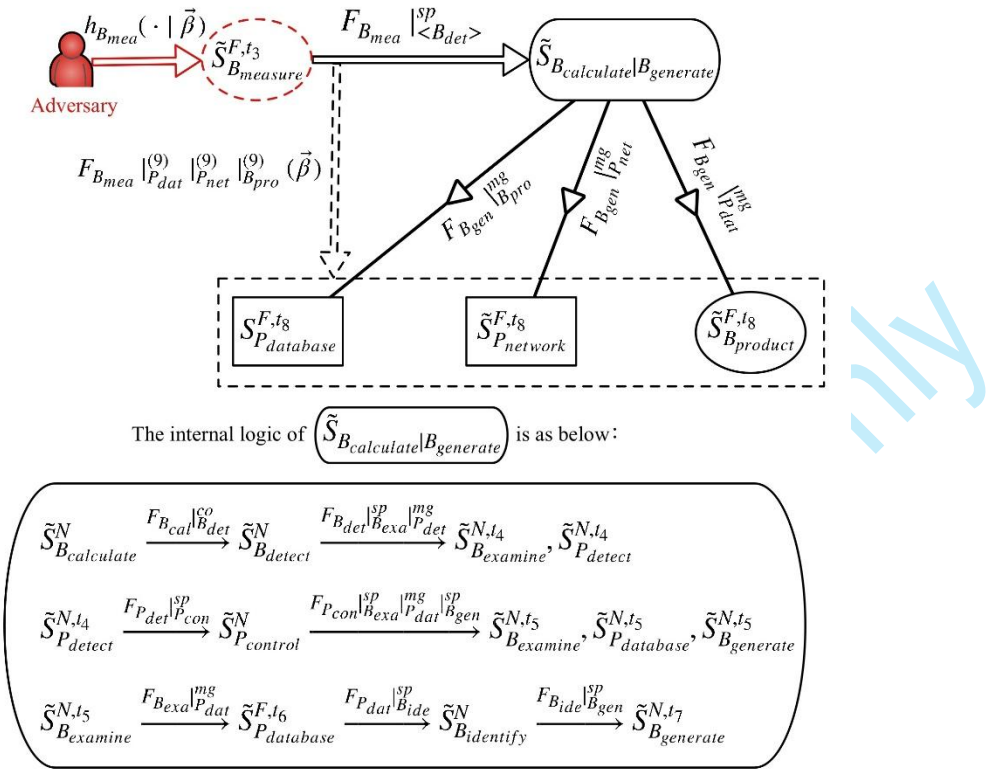


Figure 22 Mathematical expression of the ripple effect in the stage 2 of the attack

6.2.3 Stage 3 of the attack

The threat model for the stage 3 of the attack against the power architecture is shown in Figure 23. After noticing changes in the actual measurements of the attacked monitor stations, the adversary realizes that the power production business has completed the parameter modification. The adversary then continues to inject false data $\vec{\gamma}$ into the monitor stations, making the actual measurements of them start to decrease with respect to the stage 2 of the attack. This causes the estimates $\vec{x}_{\gamma}$ calculated by the state estimator to appear reduced and the residuals calculated by the bad data detector to fall below the security threshold.

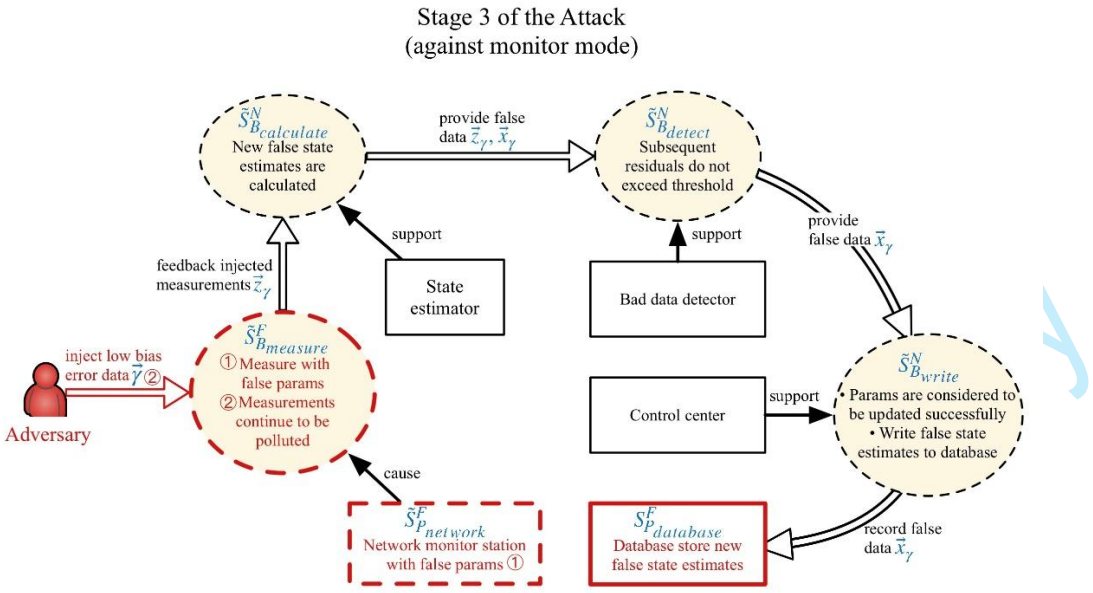


Figure 23 Threat model for the stage 3 of the power architecture attack

The control center receiving the feedback considers the adjustment to be valid, thus keeping the power production business operating with the error parameter $\vec{x}_u$. The adversary continues to inject false data so that the calculated residuals are always at a security level and never triggering an alarm. Not only does this result in erroneous data $\vec{x}_\alpha, \vec{x}_u, \vec{p}_u$ remaining in the database, and the power production business incorrectly operating for a long period of time, but it also causes the database continuing to record the erroneous state estimate $\vec{x}_\gamma$. Up to now, the adversary has been able to achieve the goal of affecting the power production business in the long term through threat ripples. The comparison table of the threat model functions is shown in Table 10, and the whole process can be expressed by the ripple functions as:

$$\begin{aligned} I_{P_{net}}|^{sp}_{B_{mea}}(\tilde{S}^{F,t_8}_{P_{net}}) &= A^{t_8}_{P_{net}}|^{sp}_{B_{mea}} \\ E_{B_{mea}}(\vec{\gamma}, A^{t_8}_{P_{net}}|^{sp}_{B_{mea}}) &= \tilde{S}^{F,t_9}_{B_{mea}} \\ F_{B_{wri}}|^{mg}_{P_{dat}} \cdot F_{B_{\det}}|^{sp}_{B_{wri}} \cdot F_{B_{cal}}|^{co}_{B_{\det}} \cdot F_{B_{mea}}|^{sp}_{<B_{\det}>}(\tilde{S}^{F,t_9}_{B_{mea}}) &= S^{F,t_{10}}_{P_{dat}} \end{aligned} \tag{72}$$

Table 10 Comparison table of the threat model functions for the stage 3 of the attack

Object	Input	Action function	State function	Output	Ripple function

$P_{network}$	$[V_{u_1}\cdots V_{u_n}, \theta_{u_1}\cdots\theta_{u_n}]$	Collecting power production data and running calculations with error parameters	Support estimation of B_{measure} using anomalous data.	Abnormal support	$I_{P_{net}}\|^{sp}_{B_{mea}}(\tilde{S}^{F,t_8}_{P_{net}}) = A^{t_8}_{P_{net}}\|^{sp}_{B_{mea}}$
B_{measure}	$\vec{\gamma}$, Abnormal support	Solve for $\vec{z}_\gamma = h(\vec{x},\vec{p}_e)+\vec{\gamma}+\vec{e}$	Provide error $\vec{z}_\gamma$ to $B_{\text{calculate}}$	$\vec{z}_\gamma$	$E_{B_{mea}}(\vec{\gamma}, A^{t_8}_{P_{net}}\|^{sp}_{B_{mea}}) = \tilde{S}^{F,t_9}_{B_{mea}}, F_{B_{mea}}\|^{sp}_{<B_{\text{det}}>}(\tilde{S}^{F,t_9}_{B_{mea}}) = \tilde{S}^{N}_{B_{cal}}$
$B_{\text{calculate}}$	$\vec{z}_\gamma$	Minimize $J=(\vec{z}_\gamma - h(\vec{x}_\gamma))^T \cdot G^{-1}\cdot(\vec{z}_\gamma - h(\vec{x}_\gamma))$ to solve for $\vec{x}_\gamma$	Provide error $\vec{z}_\gamma,\vec{x}_\gamma$ to B_{detect}	$\vec{z}_\gamma,\vec{x}_\gamma$	$F_{B_{cal}}\|^{co}_{B_{\text{det}}}(\tilde{S}^{N}_{B_{cal}}) = \tilde{S}^{N}_{B_{\text{det}}}$
B_{detect}	$\vec{z}_\gamma,\vec{x}_\gamma$	Solve for $\vec{r}_\gamma = \vec{z}_\gamma - h(\vec{x}_\gamma)$ and substitute into $r_i^N(\gamma)=\frac{\|r_i(\gamma)\|}{\sqrt{G_{ii}S_{ii}}}$	Find $r_i^N(\beta)\le\lambda$, meaning the abnormal operation remains undetected, then provide abnormal data $\vec{x}_\gamma$ to B_{write}	$\vec{x}_\gamma$	$F_{B_{\text{det}}}\|^{sp}_{B_{wri}}(\tilde{S}^{N}_{B_{\text{det}}}) = \tilde{S}^{N}_{B_{wri}}$
B_{write}	$\vec{x}_\gamma$	Consider params $\vec{p}_u,\vec{x}_u$ correct and the update successful as control center receives no alarm	Write error $\vec{x}_\gamma$ to $P_{database}$	$\vec{x}_\gamma$	$F_{B_{wri}}\|^{mg}_{P_{dat}}(\tilde{S}^{N}_{B_{wri}}) = S^{F,t_{10}}_{P_{dat}}$
$P_{database}$	$\vec{x}_\gamma$	Record error $\vec{x}_\gamma$	—	—	—

The mathematical representation of the ripple effect in the stage 3 of the attack is shown in Figure 24. The incorrectly modified vulnerable monitor stations will support B_{measure} by running calculations with error

parameters, while the adversary injects false data $\vec{\gamma}$ into them. These cause the business B_{measure} enters into a new state $\hat{S}^{t_8}_{B_{dat}}$ and ripple the threat to the platform $P_{database}$ at depth 4 in the dependency chain, thus achieving the goal of writing the newly generated error data into the database and affecting the power architecture for a long time after the power production and network parameters have been adjusted, i.e.:

$$F^{t_9}_{B_{mea}}|^{(4)}_{P_{\text{dat}}}(\vec{\gamma}, A^{t_8}_{P_{net}}|^{sp}_{B_{mea}}) = S^{F,t_{10}}_{P_{\text{dat}}} \tag{73}$$

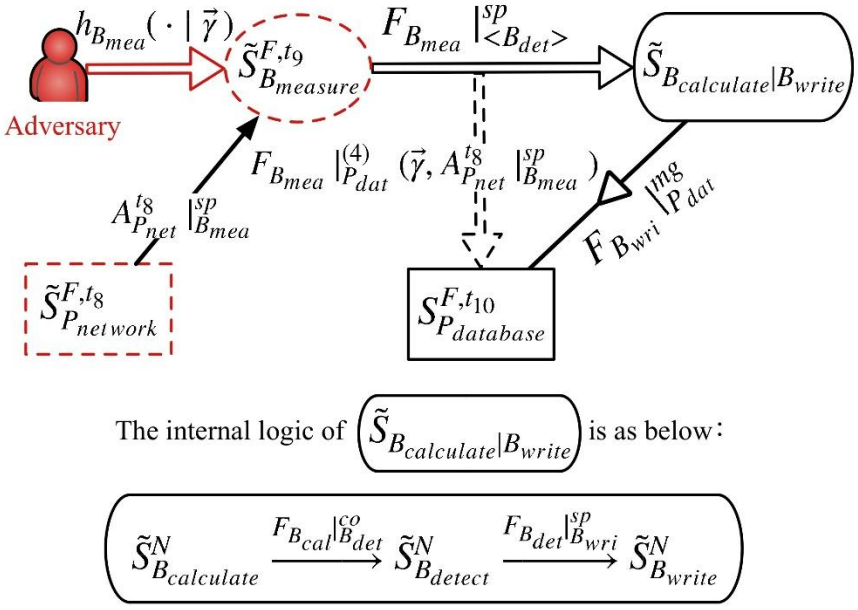


Figure 24 Mathematical expression of the ripple effect in the stage 3 of the attack

Threat modeling of the typical power production scenario utilizing the RipA model can effectively evaluate the consequences of ripple effects on the normal operation of this informatization architecture. The modeling approach in this paper enables defenders to identify the interdependencies among platforms and businesses in an informatization operation scenario. It can help to discover not only the vulnerable objects and threat ripple dependency chains that would cause significant damage once exploited by an adversary, but also the attack tactics that can cause the abnormal state to persist. The mathematical representation approach in this paper can support defenders in designing calculation algorithms oriented towards threat transmission, so as to quantify the effects of threat ripple and explore strategies to stop threat delivery and reduce its effects.

7 Model comparison

This paper builds on our previous work "Threat Ripple Model: A Model to Characterize Business-Oriented Attacks Based on Business Dependencies" [66], the threat ripple model proposed in that paper has already been compared with the influential ATT&CK model, demonstrating the advantages of the proposed modeling approach. Therefore, to further compare the capabilities of different modeling approaches in characterizing threat ripple, this paper selects the widely used attack graph model, the representative OWASP threat model adopted in industry, and the classic metamodel proposed for cyber physical system (CPS) scenarios. Although these approaches were not designed to characterize business oriented attacks or threat ripple and cannot express threats

at the business level, no existing threat model addresses such problems. Therefore, for a fair comparison, this paper still applies these approaches to characterize the transmission logic of threats among businesses.

7.1 Attack Graph Model

Figure 25 The representation of the stage 1 of the attack by the attack graph model

The attack graph model was selected as a representative model for the attack process. The attack graph modeling approach is a graph-based method for analyzing and assessing network vulnerabilities [58][59]. The stage 1 of the attack represented by the attack graph model is shown in Figure 25, we used an attack graph model to express the stage 1 of the attack. While the attack graph construction approach can accurately characterize the attack behaviors and execution steps, lacking the abstraction of the affected objects makes it difficult for defenders to evaluate which platform and business assets need reinforced. Besides, the deliberate focus on attack steps makes the attack graph model become very complicated when characterizing threats transmissions in a complex architecture with many systems, and its content grows explosively as the number of involved affected

objects increases. In contrast, the modeling approach of the RipA model is based on the architecture's own operational logic rather than attack logic, which allows threat transmissions to be characterized concisely and clearly, thereby helping defenders to prevent threats from spreading in time during a threat event.

7.2 OWASP Model

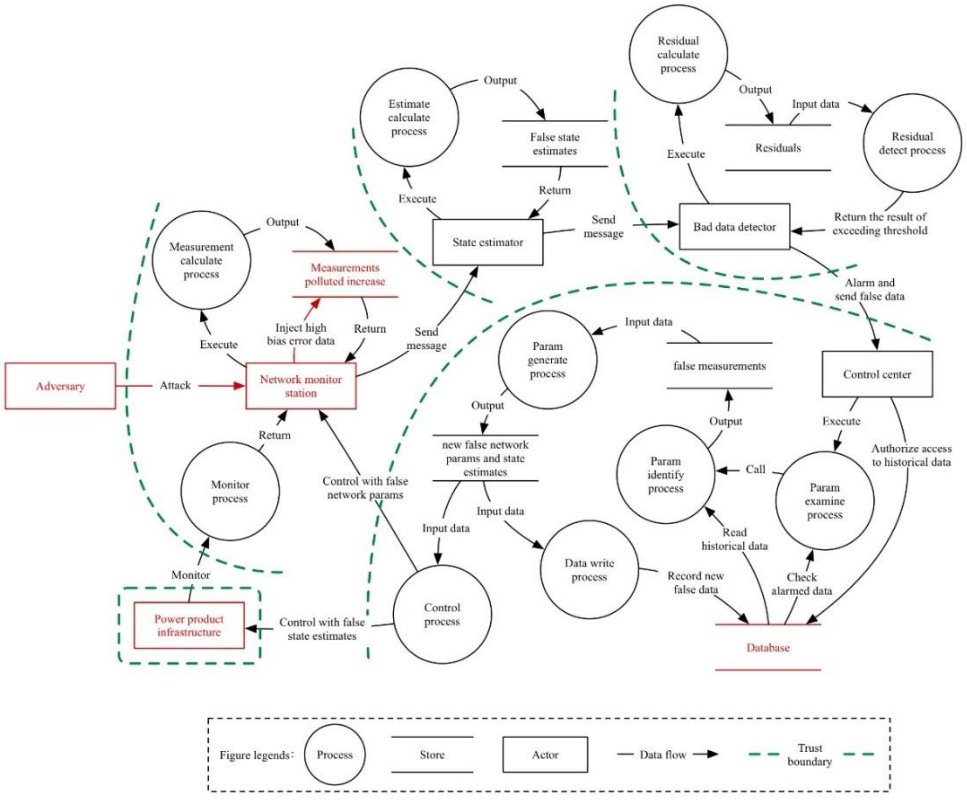


Figure 26 The representation of the stage 2 of the attack by the OWASP model

Typical object-oriented threat models, such as Microsoft's Threat Modeling Tool and the OWASP model, can express object and threat attributes (Microsoft has fixed options for threat attributes, while OWASP allows for customization). We use it to represent the stage 2 of the attack, as shown in Figure 26. This model is constructed based on execution objects, storage objects, and process objects [60]. With data flow's helping, the approach can explain the logic of threats affecting among objects. However, this model is designed for the platform rather than the business. It lacks consideration of business behaviors and interaction patterns among businesses and platforms, thus results in its expression confusing business operation logic with platform running logic. Although the OWASP model can characterize threat transmission, it cannot determine whether an affected object's abnormal status in the threat transmission chain is caused by abnormal upstream business operating or platform running. It also cannot distinguish whether the spread logic of specific threat is based on data stream or code stream. These issues are arisen from the platform perspective adopted in the model's design, which results in the modeling approach extremely detrimental to defenders' ability to identify and reinforce vulnerable business assets.

The RipA model decouples the system into the platform and the business, abstracting the information interaction into the business data stream and the code data stream. These make the defenders easily analyze the

affecting logic of different types of threats in the architecture during attack inference, and utilize the mathematical expressions to calculate the attack effect, thereby adopting effective security strategies for vulnerable platform or business assets.

7.3 Meta Model for CPS

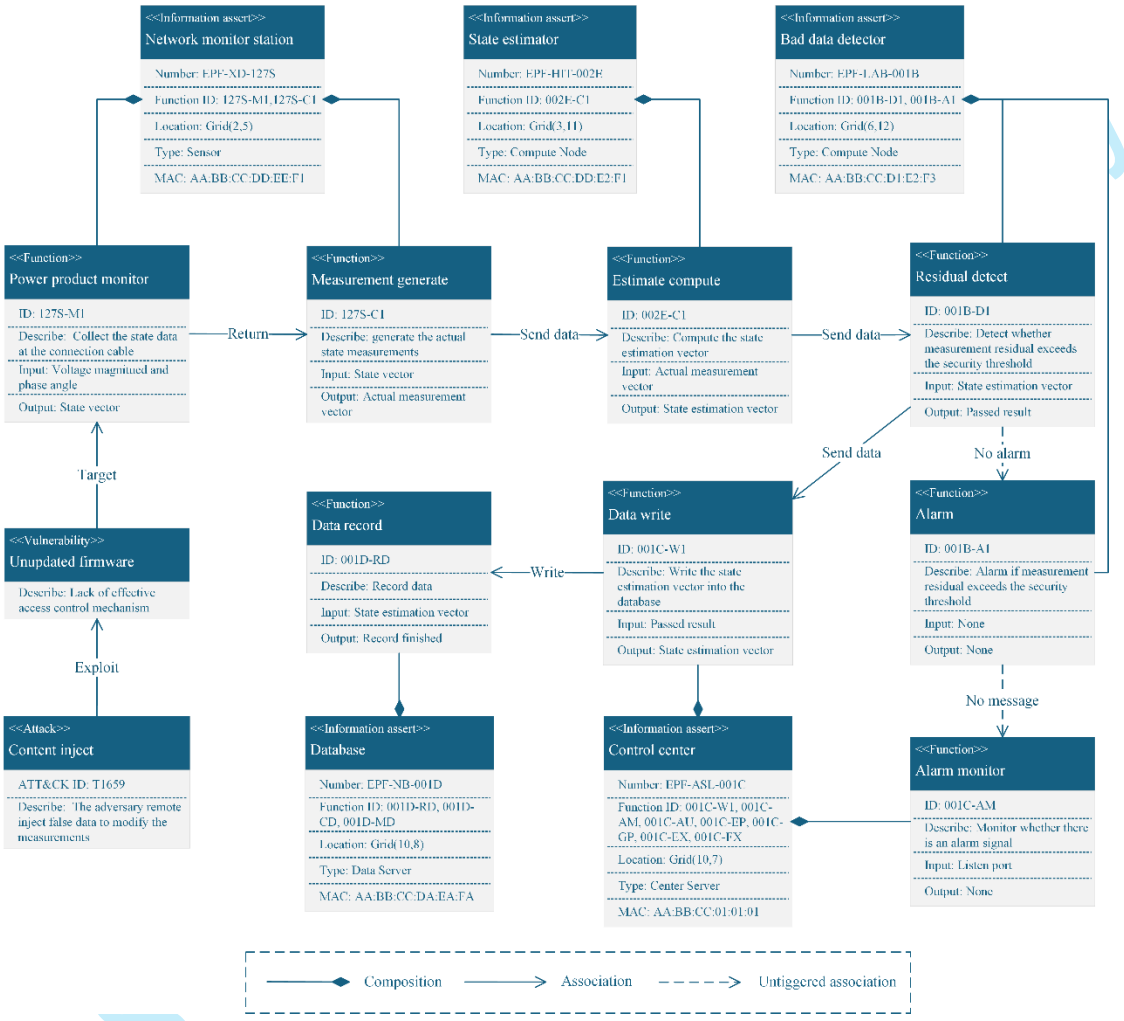


Figure 27 The representation of the stage 3 of the attack by the meta model for CPS

Meta-model is a “model of models”. It is an abstraction and description of the structure of models in a particular domain or system, defining the components, attributes, relationships, and rules of these models. In short, a meta-model defines the framework and rules for constructing other models, and is the foundation for understanding, designing, and managing complex information systems [61][62]. The paper [61] proposes a meta-model for CPS, and we use the approach described in it to express the stage 3 of the attack, as shown in Figure 27. This type of CPS meta-models can provide a detailed descriptions of infrastructure platforms' various attributes and constituent elements. Thus, many high-level CPS models based on meta-models that are adapted to different production scenarios have been designed [63][64][65]. Unfortunately, none of the CPS meta-models takes into account business attributes and constituent elements, even though many adversaries target CPS businesses. The lack of business characterization leaves defenders with no choice but to passively maintain

abnormal platforms when suffering such threats, which make it difficult to uncover the adversaries' real attack objectives and take effective countermeasures.

The CPS production environment is an information architecture that, once attacked, can easily trigger threats rippling across businesses. Current modeling approaches that do not support business characterization makes it tricky for defenders to respond effectively when no platform failure occurs, but various businesses exhibit obvious anomalies, rendering them unable to take effective measures to prevent the spread of threats. For example, in the case of the attack on the power production architecture described in this paper, the state estimator and the bad data detector only performed as specified, with no functional abnormalities or capability deficiencies, and the malicious code from the adversary did not move laterally. However, it is not only the running code that can affect the platform, but also the business data: due to false data put into the network monitoring station, threats transmit along the business dependency chain in the architecture. This ultimately results in the database being polluted and the control center making incorrect judgements and taking wrong actions on the architecture's operation. This causes the architecture to remain in an abnormal production state.

7.4 Advantages of the RipA Model

Strictly speaking, the RipA model comprises three layers: the architecture model layer, the attack and threat characterization layer, and the ripple effect representation layer: the architecture model is abstracted from the threat scenario. Attack and threat characterization layer is constructed according to the interacting objects and operational mechanisms defined in the architecture model. On this basis, the ripple effect representation layer presents the final depth and consequences of ripple effects. Each layer also has its corresponding mathematical representation, as shown in Figure 28.

Table 11 Comparison of Typical Threat Models

Model	Modeling perspective	Modeling object	Business character-rization	Abstraction flexibility	Threat transmission expression	Quantitative computation support	Complexity
Attack Graph	Attack Steps	Vulnerability & Attack Steps	×	Low	Weak	√	High
OWASP	Platform-Oriented	Platforms & Information stream	×	High	Medium	×	Medium
CPS Meta-Model	System Structure	Devices	×	Low	Weak	×	High
RipA	component Dependency	Platform & Business	√	High	Strong	√	Medium

These characteristics place the RipA model in a different dimension of capability from threat models that mainly focus on platform running and code execution, including the attack graph model

that graphically represents attack paths, the representative OWASP model designed around objects, and the CPS metamodel developed from system structure, as shown in Table 11. The comparison results show that existing threat models can describe attack behavior and threat transmission only at the levels of platform running and code execution. Therefore, they are not comparable with the RipA model in characterizing business oriented attacks or representing threat ripple among businesses.

Moreover, in terms of modeling perspective, the RipA model adopts a business perspective that none of the other models consider. Without operational logic, these models are inherently unable to represent the stage in which effects transmit after an attack.

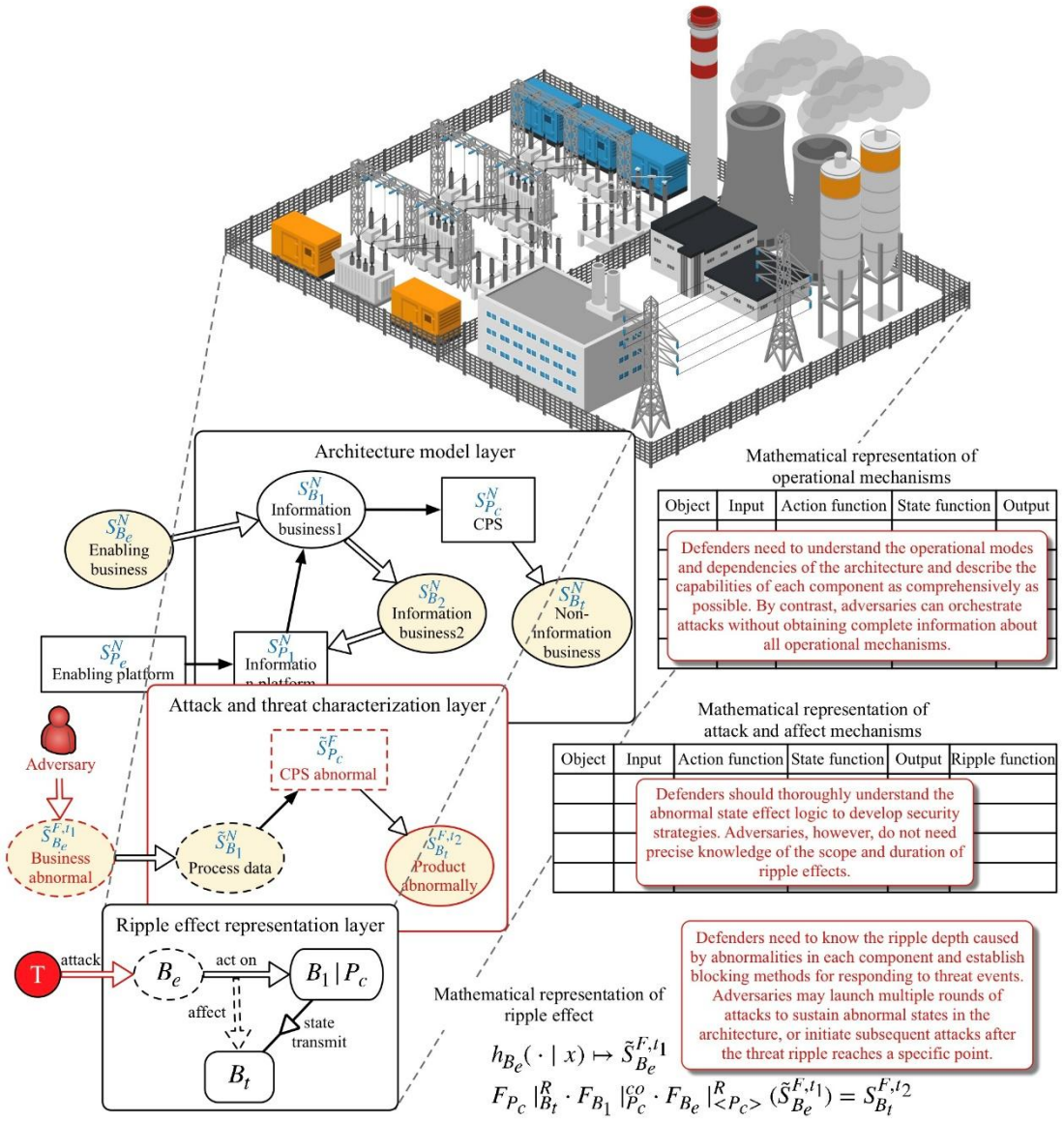


Figure 28 The three-layer architecture and mathematical expression of RipA model

The RipA model provides stronger explanatory capability in characterizing business oriented attacks and threat ripple. Its value in supporting defense is mainly reflected in three aspects:

1.Rather than characterizing an attack directly from the attack behavior itself, the model makes explicit the dependencies that are implicit in the operational processes of an architecture. This enables defenders to understand the threat ripple process during modeling and identify the actual purpose for which an adversary selects a specific object as the attack target.

2.By explaining threat ripple through state effect logic, the domino effect of threat transmission is decomposed into a sequence of stages involving threat input, processing, state update, anomaly emergence, and threat output for each component. Threat transmission thus becomes an interpretable process with explicit causal relationships.

3.It provides an expressive foundation for instantiable threat computation, making the assessment of threat spread no longer limited to qualitative judgments about attack consequences. Accordingly, the affected scope, ripple depth, and blocking points of threat ripple can be evaluated and calculated in conjunction with the business operation and platform running modes of the architecture itself.

8 Discussion

This paper presents a theoretical study on methodology whose core value lies in establishing a systematic theory of threat ripple. It provides a theoretical foundation for researchers to further refine the model and run simulations, and for defenders to construct algorithms and perform assessments based on actual operational mechanisms, or analyze threats in their own scenarios using the proposed methodological framework.

Compared with the authors' previous research on information architecture scenarios in paper [66], the incremental contributions of this paper do not simply involve applying the original model to new cases. Instead, this paper extends the previous research in three aspects: theoretical scope, threat transmission mechanisms, and support for threat computation:

Theoretical scope: This paper extends the scope of threat ripple research from platforms and information businesses within information systems to informatization architectures that also contain non-information businesses. It further demonstrates that cyber attacks can affect not only platform running and information business processing, but also mission execution in physical space through state feedback, business workflows, and coordination across components.

Threat transmission mechanisms: This paper no longer explains threat transmission among platforms and businesses solely through dependency sequences. Instead, it further distinguishes management dependencies and support dependencies formed directly or indirectly, as well as state dependencies arising from multiple dependencies and coordination patterns. It then discusses in detail the state transmission mechanisms through which effects spread after an attack. These enable the RipA model to explain how abnormal states are formed, how threats penetrate intermediate components, when ripple effects terminate, and why a component at the same point may produce different ripple depths along downstream relations when it is in different states.

Support for threat computation: The mathematical representation designed in this paper can support threat ripple computation and prevent the modeling approach from remaining only at the level of formal representation. The attack function describes the effect of external attack inputs on a target component. The state function expresses the information output by a component in a specific state.

The action function characterizes the new state formed by a component under the effect of input information. The ripple function describes the pattern of threat spread. These functions enable the model to explain the transmission conditions and causal relationships of threat ripple.

Logically, the computational rules underlying the mathematical expressions presented in this paper can be regarded as an algorithm. By assigning several parameters and specifying the calculation process, the framework can support general testing by academic researchers and defenders in common scenarios. This paper could readily assign numerical values and conduct algorithmic simulations to demonstrate the computability of the proposed approach. It would not be difficult, but would have little practical significance. Instead, this paper still recommends that researchers construct parameters according to the problems they face, and that defenders construct parameters according to the specific conditions of their scenarios or use the mathematical representation framework to conduct threat analysis based on actual operational mechanisms. This is because, in actual attack and defense confrontations, no fixed computational method for characterizing threat ripple is applicable to all situations:

- The operation of an architecture is dynamic. Even within the same scenario, its operation mode may vary over time when responding to different situations and challenges. For example, a metro transportation architecture may adopt dedicated execution plans under normal conditions, high pressure conditions, and terrorist attacks, thereby changing its business operation logic and the state dependency mechanisms among components.
- Architectures of the same type may also differ in their operational mechanisms depending on the country, region, and available resources. For example, architectures for electric power generation, transmission, and distribution may differ across regions in energy structure, power dispatch rules, and automation levels. Consequently, even when the same type of component enters an abnormal state, the resulting threat transmission paths, the conditions under which the states of the involved components change, and the scope of effect spread may differ.

Therefore, it is clearly inappropriate to assign static numerical values to threats and risks without considering specific scenarios, or to attempt to directly characterize all threat ripple processes using a fixed algorithm. A more appropriate method is for researchers to modify the model, construct algorithms, and run simulations for specific problems based on the proposed theory and mathematical representation approach. Operators, meanwhile, should define the execution logic and state thresholds of components according to the operational rules of their scenarios and the actual capabilities of individual components, instantiate the computational functions within the framework. During attack inference or vulnerability discovery, threat information can be input into the designated target. The triggering conditions for state changes in each component and the logic of threat spread can then be calculated.

The unavoidability of threat ripple is essentially a conditional transmission characteristic jointly determined by the operational mechanisms of the architecture itself and the conditions that trigger state changes in specific components. The purpose of developing mathematical expressions for threat ripple is not to prescribe a fixed calculation approach for all scenarios, but to make the causal relationships among threat action, state evolution, and effect spread computable. This ensures that the RipA threat model possesses general analytical capability across scenarios while remaining adaptable to differences among specific scenarios.

9 Conclusion

With the rapid advancement of digitalization, information systems and physical systems are accelerating their convergence, placing higher demands on threat modeling for informatization architectures. The modeling approach proposed in this paper not only characterizes attack processes at the business level but also expresses the domino effect in which threats transmit and expand their effects through platform running and business operation mechanisms within the architecture. By introducing mathematical formulations, the RipA model supports quantitative threat computation and analysis in attack scenarios, thereby assisting security researchers in assessing potential threats to architecture and rapidly contain possible ripple effects when encountering an attack.

As cyberspace, physical space, and cognitive space continue to converge, adversaries are increasingly likely to focus on leveraging attacks in cyberspace to transmit threats into other domains, thereby expanding the scope of their effects. The threat modeling approach presented in this paper is applicable to the study of such cross-domain threat effect patterns. Future work could further investigate the specific types of threat effects on platforms and businesses and develop supporting tools to facilitate the analysis of threats in informatization architectures.

Acknowledgments

We thank the anonymous reviewers for their helpful comments.

Funding

This work was supported by the National Key Research and Development Program of China under Grant 2025YFB3109803 and Heilongjiang Provincial Natural Science Foundation of China under Grant JQ2024F001.

Conflicts of interest

The authors declare that they have no conflict of interest.

Data availability statement

No data are associated with this article.

Author's Contributions

Shiliang Ao and Binxing Fang jointly contributed to the threat modelling methodology, mathematical analysis, attack inference, and manuscript revision. Shiliang Ao proofread the manuscript, and Binxing Fang provided resource support for this paper.

Reference

[1] T. Yadav, and A. M. Rao, ”Technical Aspects of Cyber Kill Chain.” Third International Symposium on Security in Computing and Communications (SSCC’15), 2015. (CP)

[2] B. E. Strom, A. Applebaum, D. P. Miller, K. C. Nickels, A. G. Pennington, and C. B. Thomas, MITRE ATT&CK™ : Design and Philosophy. Technical report. The MITRE Corporation, 2018. (CP)

[3] Falliere, Nicolas, Liam O. Murchu, and Eric Chien. "W32. stuxnet dossier." White paper, symantec corp., security response 5.6 (2011): 29. (CP)

[4] McDonald, Geoff, et al. "Stuxnet 0.5: The missing link." Symantec Report 26 (2013). (CP)

[5] Alwi, A., & Ariffin, K. A. Z. (2018, November). Information Security Risk Assessment for the Malaysian Aeronautical Information Management System. In 2018 Cyber Resilience Conference (CRC) (pp. 1-4). IEEE.

[6] Sakhabieva, G. A. (2016). The new generation of informatization systems overview. Международный научно-исследовательский журнал, (6-1 (48)), 85-87.

[7] Lallie, H. S., Debattista, K., & Bal, J. (2020). A review of attack graph and attack tree visual syntax in cyber security. Computer Science Review, 35, 100219.

[8] Aksu, M. U., Dilek, M. H., Tatlı, E. İ., Bicakci, K., Dirik, H. I., Demirezen, M. U., & Aykır, T. (2017, October). A quantitative CVSS-based cyber security risk assessment methodology for IT systems. In 2017 International Carnahan Conference on Security Technology (ICCST) (pp. 1-8). IEEE.

[9] Senarak, C. (2024). Port cyberattacks from 2011 to 2023: a literature review and discussion of selected cases. *Maritime Economics & Logistics*, *26*(1), 105-130.

[10] Zeng, J., Wu, S., Chen, Y., Zeng, R., & Wu, C. (2019). Survey of attack graph analysis methods from the perspective of data and knowledge processing. Security and Communication Networks, 2019(1), 2031063.

[11] Ivanov, D., Dolgui, A., & Sokolov, B. (2025). *Handbook of Ripple Effects in the Supply Chain*. Springer.

[12] D.J. Bodeau, C. D. McCollum, and D. B. Fox, Cyber threat modeling: Survey, assessment, and representative framework. MITRE CORP MCLEAN VA MCLEAN, 2018. (CP)

[13] Dusane, P. S., & Pavithra, Y. (2020). Logic bomb: an insider attack. Int J, 9(3).

[14] V.Nagaraju, L. Fiondella, and T. Wandji, ”A survey of fault and attack tree modeling and analysis for cyber risk management.” 2017 IEEE International Symposium on Technologies for Homeland Security (HST) IEEE, 2017. (CP)

[15] L. Zhang, A. Taal, R. Cushing, C. de Laat, and P. Grosso, A risk-level assessment system based on the STRIDE/DREAD model for digital data marketplaces. International Journal of Information Security, 2022, 21(3), 509-525. (CP)

[16] Hovmark, O., & Schüldt, E. (2020). Towards Extending Probabilistic Attack Graphs with Forensic Evidence: An investigation of property list files in macOS.

[17] R. A. Caralli, J. F. Stevens, L. R. Young, and W. R. Wilson, Introducing OCTAVE Allegro: Improving the Information Security Risk Assessment Process, 2007. (CP)

[18] Helou M, Weinstein E S, Kalaji J, et al. Pager Explosion in Beirut: An Unprecedented Event[J]. Disaster Medicine and Public Health Preparedness, 2024, 18: e215.

[19] I. Kotenko, and E. Doynikova, The CAPEC based generator of attack scenarios for network security evaluation. IEEE International Conference on Intelligent Data Acquisition & Advanced Computing Systems: Technology & Applications, 2015, pp.436-441. (CP)

[20] J. Wynn, Threat assessment and remediation analysis (tara). MITRE CORP BEDFORD MA BEDFORD United States, 2014. (CP)

[21] D. Wichers, Owasp top-10 2013. OWASP Foundation, February, 2013. (CP)

[22] N. Shevchenko, T. A. Chick, P. O’Riordan, T. P. Scanlon, and C. Woody, Threat modeling: a summary of available methods. Carnegie Mellon University Software Engineering Institute Pittsburgh United States, 2018. (CP)

[23] STIX: Assets Affected in an Incident. Available online:

http://stixproject.github.io/documentation/idioms/affected-assets/ (accessed on 5 May 2018). (CP)

[24] S. Kotusev, Togaf-based enterprise architecture practice: an exploratory case study. Communications of the Association for Information Systems, 2018, 43(1), 321-359. (CP)

[25] Z. Tao, Y, Luo, C. Chen, M. Wang, and F. Ni, Enterprise application architecture development based on DoDAF and TOGAF. Enterprise Information Systems, vol. 11, Jul. 2015, p. 627-651. (CP)

[26] Harry, C. T., & Gallagher, N. (2022). Effects-Centric Approach to Assessing Cybersecurity Risk. Center for International & Security Studies, U. Maryland.

[27] Kure, H. I., & Islam, S. (2019). Assets focus risk management framework for critical infrastructure cybersecurity risk management. IET Cyber-Physical Systems: Theory & Applications, 4(4), 332-340.

[28] Katsumata, P., Hemenway, J., & Gavins, W. (2010, October). Cybersecurity risk management. In 2010-MILCOM 2010 Military Communications Conference (pp. 890-895). IEEE.

[29] LaLone, N. (2024). On the growing importance of routine cybersecurity: The Oldsmar Water Plant "Hack". In Case Studies in Disaster Response (pp. 237-248). Butterworth-Heinemann.

[30] Lippmann, R. P., & Riordan, J. F. (2016). Threat-based risk assessment for enterprise networks. Lincoln Lab. J, 22(1), 33-45.

[31] Russo, P., Caponi, A., Leuti, M., & Bianchi, G. (2019). A web platform for integrated vulnerability assessment and cyber risk management. Information, 10(7), 242.

[32] George, A. S. (2024). When Trust Fails: Examining Systemic Risk in the Digital Economy from the 2024 CrowdStrike Outage. Partners Universal Multidisciplinary Research Journal, 1(2), 134-152.

[33] ben Othmane, L., Weffers, H., & Klabbers, M. (2013, July). Using attacker capabilities and motivations in estimating security risk. In Workshop on risk perception in it security and privacy, Newcastle, UK.

[34] Pollard M. A Case Study of Russian Cyber-Attacks on the Ukrainian Power Grid: Implications and Best Practices for the United States[J]. Pepperdine Policy Review, 2024, 16(1).

[35] Mohurle, S., & Patil, M. (2017). A brief study of wannacry threat: Ransomware attack 2017. International journal of advanced research in computer science, 8(5), 1938-1940.

[36] Beerman, J., Berent, D., Falter, Z., & Bhunia, S. (2023, May). A review of colonial pipeline ransomware attack. In 2023 IEEE/ACM 23rd International Symposium on Cluster, Cloud and Internet Computing Workshops (CCGridW) (pp. 8-15). IEEE.

[37] Čule, Z. (2024). Kibernetička sigurnost-aktualnosti u zaštiti podataka i regulativni okvir (Master's thesis, Sveučilište u Splitu, Sveučilište u Splitu, Fakultet za forenzičke znanosti bez pravne osobnosti).

[38] Yu, Z., Wang, H., & Chen, F. (2023). Security of railway control systems: A survey, research issues and challenges. High-speed Railway, 1(1), 6-17.

[39] Tan, K., & Walters, K. (2026). The Butterfly Effect. *Children & Schools*, *48*(3), 135-139.

[40] INCOSE (Ed.). (2023). INCOSE systems engineering handbook. John Wiley & Sons,8-10.

[41] Cui, H., Hong, J., & Louden, R. (2024). An Overview of the Security of Programmable Logic Controllers in Industrial Control Systems. Encyclopedia, 4(2), 874-887.

[42] Browning, J. G., & Tuma, S. (2015). If your heart skips a beat, it may have been hacked: cybersecurity concerns with implanted medical devices. SCL Rev., 67, 637.

[43] Lins, M., Mayrhofer, R., Roland, M., Hofer, D., & Schwaighofer, M. (2024). On the critical path to implant backdoors and the effectiveness of potential mitigation techniques: Early learnings from XZ. arXiv

preprint arXiv:2404.08987.

[44] Sheahan, M., C. Steitz, and A. Rinke. "Satellite outage knocks out thousands of Enercon's wind turbines." Reuters. Reuters (2022).

[45] Hoffmann, R., Napiórkowski, J., Protasowicki, T., & Stanik, J. (2020). Risk based approach in scope of cybersecurity threats and requirements. Procedia Manufacturing, 44, 655-662.

[46] Połeć, J., & Trzaskowski, W. (2022). The Influence of Cyberwars on Socioeconomic Activity of Residents of Central and Eastern Europe. Przegląd Nauk o Obronności, 2021(12), 111-130.

[47] Kothari, S., & Joshi, S. (2020, October). Analysis of Android Applications to Detect Botnet Attacks. In 2020 International Conference on Smart Innovations in Design, Environment, Management, Planning and Computing (ICSIDEMPC) (pp. 144-150). IEEE.

[48] Marion, J. Y. (2023). Rançongiciel, une plongée dans le monde de la cybercriminalité. The Conversation France.

[49] Sullivan, R. J. (2014). Controlling security risk and fraud in payment systems. Federal Reserve Bank of Kansas City, Economic Review, 99(3), 47-78.

[50] Mirsky, Y., & Guri, M. (2020). Ddos attacks on 9-1-1 emergency services. IEEE Transactions on Dependable and Secure Computing, 18(6), 2767-2786.

[51] Monticelli, A. (2000). Electric power system state estimation. Proceedings of the IEEE, 88(2), 262-282.

[52] Zhu, J., & Abur, A. (2006). Identification of network parameter errors. IEEE Transactions on Power Systems, 21(2), 586-592.

[53] Andersson, G., Donalek, P., Farmer, R., Hatziargyriou, N., Kamwa, I., Kundur, P., ... & Vittal, V. (2005). Causes of the 2003 major grid blackouts in North America and Europe, and recommended means to improve system dynamic performance. IEEE transactions on Power Systems, 20(4), 1922-1928.

[54] Castillo, M. R., London, J. B., Bretas, N. G., Lefebvre, S., Prévost, J., & Lambert, B. (2010). Offline detection, identification, and correction of branch parameter errors based on several measurement snapshots. IEEE Transactions on power systems, 26(2), 870-877.

[55] Lin, Y., & Abur, A. (2018). Robust state estimation against measurement and network parameter errors. IEEE Transactions on power systems, 33(5), 4751-4759.

[56] Liu, C., He, W., Deng, R., Tian, Y. C., & Du, W. (2022). False-data-injection-enabled network parameter modifications in power systems: Attack and detection. IEEE Transactions on Industrial Informatics, 19(1), 177-188.

[57] Zhao, J., & Mili, L. (2018). Vulnerability of the largest normalized residual statistical test to leverage points. IEEE Transactions on Power Systems, 33(4), 4643-4646.

[58] Wachter, J. (2023). Graph models for Cybersecurity--A Survey. arXiv preprint arXiv:2311.10050.

[59] Tayouri, D., Baum, N., Shabtai, A., & Puzis, R. (2023). A survey of MulVAL extensions and their attack scenarios coverage. IEEE Access, 11, 27974-27991.

[60] Conklin, L., Drake, V., Strittmatter, S., Braiterman, Z., & Shostack, A. (2023). Threat modeling process. OWASP. org link: https://owasp. org/www-community/Threat_Modeling_Process# stride-threat--mitigation-techniques.

[61] Martins, G., Bhatia, S., Koutsoukos, X., Stouffer, K., Tang, C., & Candell, R. (2015, August). Towards a systematic threat modeling approach for cyber-physical systems. In 2015 Resilience Week (RWS) (pp. 1-6). IEEE.

[62] Fernandez, E. B. (2016, August). Threat modeling in cyber-physical systems. In 2016 IEEE 14th Intl Conf on Dependable, Autonomic and Secure Computing, 14th Intl Conf on Pervasive Intelligence and Computing, 2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech) (pp. 448-453). IEEE.

[63] Khan, R., McLaughlin, K., Laverty, D., & Sezer, S. (2017, September). STRIDE-based threat modeling for cyber-physical systems. In 2017 IEEE PES innovative smart grid technologies conference Europe (ISGT-Europe) (pp. 1-6). IEEE.

[64] Khalil, S. M., Bahsi, H., Ochieng'Dola, H., Korõtko, T., McLaughlin, K., & Kotkas, V. (2023). Threat modeling of cyber-physical systems-a case study of a microgrid system. Computers & Security, 124, 102950.

[65] Jbair, M., Ahmad, B., Maple, C., & Harrison, R. (2022). Threat modelling for industrial cyber physical systems in the era of smart manufacturing. Computers in Industry, 137, 103611.

[66] Ao, S., Fang, B., Xiao, X., & Zhang, H. (2025). Threat ripple model: A model to characterize business-oriented attacks based on business dependencies. *Security and Safety*, *4*, 2025003.

[67] Amazon, A. W. S. (2017). *Summary of the amazon s3 service disruption in the northern virginia (us-east-1) region*.

[68] Thomas, L. (2023). Ripple effect: Disentangling the global impact web of US monetary policy. *Finance Research Letters*, *58*, 104431.

[69] Creel, L. (2003). Ripple effects: population and coastal regions.

[70] Dolgui, A., & Ivanov, D. (2021). Ripple effect and supply chain disruption management: new trends and research directions. *International Journal of Production Research*, *59*(1), 102-109.

[71] Newman, M. E. (2003). Ego-centered networks and the ripple effect. *Social Networks*, *25*(1), 83-95.

[72] Maddux, W. W., & Yuki, M. (2006). The "ripple effect": Cultural differences in perceptions of the consequences of events. *Personality and social psychology bulletin*, *32*(5), 669-683.

[73] Dolgui, A., Ivanov, D., & Sokolov, B. (2018). Ripple effect in the supply chain: an analysis and recent literature. *International journal of production research*, *56*(1-2), 414-430.

[74] Gajdzik, B. (2022). How steel mills transform into smart mills: Digital changes and development determinants in the Polish steel industry.

Author's Information

Shiliang Ao received the B.S. degree from Xidian University, Xi'an, China, in 2015. He is currently a Ph.D. candidate in the School of Cyberspace Science from Harbin Institute of Technology (HlT). His research interests include cybersecurity and system engineering.

Binxing Fang received the Ph.D. degree in computer science and technology from Harbin Institute of Technology, Harbin, China, in 1989. He is a member of the Chinese Academy of Engineering. His current research interests include computer networks, information and network security.