Modeling Network Security using Colored Petri Nets Model

Abdelali EL BOUCHTI, Abdelkrim HAQIQ
Computer, Networks, Mobility and Modeling Laboratory
e-NGN Research Group, Africa and Middle East
FST, Hassan 1st University, Settat, Morocco
{a.elbouchti, ahaqiq}@gmail.com

ABSTRACT**:** *Network security is a complex and challenging problem. The area of network defense mechanism design is receiving immense attention from the research community. However, the network security problem is far from completely solved. In this context, several modeling approaches have been developed, such as approaches based on attack trees (AT). Researchers have been exploring the applicability of colored Petri nets approaches to address the network security issues and some of these approaches look promising.*

Petri Nets provide a graphical notation for modeling systems and performing analysis. Colored Petri Nets (CoPNets) combine the strengths of ordinary Petri Nets with a high level programming language, making them more suitable for modeling large systems. A CoPNet model is an executable representation of a system that can be analyzed through simulation. CoPNet models are built using CoPNet Tools, a graphical software tool and interface used to create, edit, simulate, and analyze models.

This paper proposes Colored Petri Net (CoPNet) modeling approach by extending the attack trees with new modeling constructs and analysis approaches. CoPNet based attack model is flexible enough to model Internet intrusion, including the static and dynamic features of the intrusion. The process and rules of building CoPNet based attack model from AT are also presented. In order to evaluate the risk of intrusion, some cost elements are added to CoPNet based attack modeling. We show how attack trees can be converted and analyzed in CoPNets. Finally, we provide three case studies that illustrate the CoPNet approach.

Keywords: Network Security, Cyber-attack, Vulnerability, CoPNet, Attack Modeling, Attack Ggraph, Attack Tree

Received: 30 January 2014, Revised 23 March 2014, Accepted 12 April 2014

1. Introduction

A secure computer system provides guarantees regarding the confidentiality, integrity and availability of its objects (such as data, processes or services). However, systems generally contain design and implementation flaws that result in security vulnerabilities. An intrusion takes place when an attacker or group of attackers exploit security vulnerabilities and thus violate the confidentiality, integrity, or availability guarantees of a system or a network.

Intrusion Detection Systems (IDSs) [15] detect some set of intrusions and execute some predetermined action when an intrusion is detected.

Recent incidents in cyberspace [38, 13, 35] prove that network attacks can cause huge amounts of loss to governments, private enterprises, and the general public in terms of money, data confidentiality, and reputation. The research community has been paying attention to the network security problem for more than two decades. However, the problem is far from being completely solved. We frequently see a race between the security specialists and the attackers in the following sense: one day an intelligent solution is proposed to fix a network vulnerability, and the next day the attackers come up with a smarter way to circumvent the proposed countermeasure. The most important factor which makes this problem difficult is that the local network, which needs to be secured, is typically connected to the Internet and major parts of the Internet are beyond the control of network administrators. However, the Internet has become an integral component of running the daily business of government, financial institutions, and the general public. As a result, there is a pressing need to design countermeasures for network attacks.

Traditionally, network security solutions employ either protective devices such as firewalls or reactive devices such as Intrusion Detection Systems (IDSs) and both of them are used in conjunction. The intrusion detection algorithms are either based on identifying an attack signature or detecting the anomalous behavior of the system. Once an attack is detected the employed IDS notifies the network administrator who then takes an action to stop or mitigate the attack.

However, currently IDSs are not very sophisticated and they rely on ad-hoc schemes and experimental work. The current IDS technology may prove sufficient for defending against casual attackers using well known techniques, but there is still a need to design tools to defend against sophisticated and well organized adversaries.

The weakness of the traditional network security solutions is that they lack a quantitative decision framework. To this end, a few groups of researchers have started advocating the utilization of game theoretic approaches. As game theory deals with problems where multiple players with contradictory objectives compete with each other.

1.1 Scope of this Paper

The purpose of our proposed approach, called Colored Petri Net Attack Modeling approach (CoPNet) [22] is to provide intuitive modeling approach for modeling attacker behavior in vulnerable systems from security perspective, based on the concepts of attack trees and modeling abilities of Petri Nets. Some cost elements are added to CoPNet based attack modeling to evaluate the risk of intrusion. We choose Border Gateway Protocol (BGP) [23], Supervisory Control and Data Acquisition (SCADA) [27] networks and malicious insider attack (MIA) [18] as three case studies that illustrates the CoPNet approach.

Our choice of a CoPNet formalism to address the design of security policies is motivated by the following reasons: Petri Nets are well known for their graphical and analytical capabilities for the specification and verification of concurrent, synchronous, distributed, parallel, and nondeterministic systems. Various features contributing to such a success include graphical nature, the simplicity of the model, and the firm mathematical foundation. It also provides modularity in design. Hence, a Petri-net-based policy is more flexible when it is embedded into a system.

1.2 Outline of this Paper

The remainder of this paper is organized as follows. Section 2 investigates some related works. An overview of attack graph and attack tree is presented in Section 3. In Section 4, we present and define Petri Nets and CoPNet. In Section 5, we show how to build CoPNet attack model from attack tree. We show the extended CoPNet model in Section 6. In Section 7, we describe and illustrate CoPNet approach by using case studies (BGP attack, SCADA attack and malicious insider attack). Finally, we conclude the paper and give an overview of future work in Section 8.

2. Related Work

Some literatures show a comprehensive taxonomy of internet attack [4, 6]. Other common intrusion database such as [5] also creates a common namespace for all vulnerabilities and exploits. Taxonomy of attacks fails to formally express their dynamic properties. Some graph-based attack models also provide means for modeling intrusion [7]. Other research [6] uses the software fault tree approach to analyze the design and implementation of intrusion detection system. Schneier [2] was the first one to associate the term "*attack tree*" with the use of fault tree for attack modeling which made this approach more widely known. This modeling tool has proved to be simple, easy to use and easy to analyze results, and yet powerful in its modeling capability.

Besides modeling attacker behavior ATs are found to be useful for modeling system vulnerabilities and points of access. However, the capabilities of ATs are limited, because of their limited construct set and static nature.

Although cyber-attacks are common most modeling approaches do not utilize real-world attack data. Instead, researchers often use either simple cases to illustrate models, or they build generic models based on the available information. Using real-world systems and attacks in developed models would be extremely beneficial to capture strengths and weaknesses of any attack modeling approach. Our approach would be more meaningful if it could be applied to an actual vulnerable system. Another research effort that involves cyber security focuses on interaction between physical and cyber worlds during attack. In this area, researchers study consequences of cyber-attacks on some critical infrastructures such as power plant system. Petri nets could prove useful in this scenario as a tool that is capable of modeling both physical and cyber worlds, and their interactions. Our effort uses Petri Net constructs to augment and extend existing principles that are already proven useful in ATs. Although some Petri Net and CoPNet based models have existed, they are only used to model the intrusion detection system itself [8].

3. Modeling Approaches

Generally, approaches for attack modeling can be divided into two kinds: graph structure and tree structure. The tree structure use a tree to present the situation of vulnerabilities exploited to attack. Approaches based on those two structures are presented in the following.

3.1 Attack modeling

A typical attack contains the following elements:

- **Objects attacked.** These objects belong to the victims or can be regarded as public resource, such as networking bandwidth.
- **Attacker.** These objects contain the hacker's information, attacking tools and other states of attacker.
- **Attack processes.** The stages of an attack and attack processes are used to depict the attacking action.
- **Control actions.** These actions can be classified into response actions and defensive action. The former will be fired if the intrusion is detected by the system. While the latter is the controlling flow used to prevent intrusion happening.

An ideal attack model should describe all these features. But currently no approach can completely attain this goal. Among the present attack modeling methods, attack tree is the most popular one. Attack tree can capture the steps of an attack and its interdependencies. Attack tree are also used to represent and calculate probabilities, risks, cost or other weightings. The main building blocks of attack tree are nodes. Every node is used to model one step of an attack or one attacker's action. Every tree has a single root node that represents the ultimate goal of an attack. Tree hierarchy models the temporal logics of stages and goals. In bottom-up attack tree, child node means attack stage has to be successfully performed before another step occurs. AND or OR logical gate can be set to each node. A node with OR gate occurs only when any of its child node occurs. For occurrence of a node with AND gate, all of its child events are necessary. Node can be set with a value or probability. This kind of attack tree is defined as weighted attack tree. Similarly, node with value can also be defined as valued node. In this way attack tree can model cost feature of an attack and can be used to perform risk assessment.

But control actions cannot be modeled with attack trees. So there are some limits to the application of attack tree to intrusion response. Some extended model should be used to depict this important aspect of intrusion detection and response. Color petri is such an approach.

3.2 Attack graph

Attack graph is suitable and effective to describe the sequence of the vulnerabilities exploited. In recent years, many research institutes have devoted time and energy into this field, such as Carnegie Mellon University and George Mason University [16]. Attack graphs are used to determine if designated goal states can be reached by attackers attempting to penetrate computer networks from initial starting states. For this use, they are graphs in which the starting node represents an attacker at a specified network location. Nodes and arcs represent actions the attacker takes and change in the network state caused by these actions. Actions typically involve exploits or exploit steps that take advantage of vulnerabilities in software or protocols. A full Attack graph will show all possible sequences of attackers' actions that eventually lead to the desired level of privilege on the target. Figure 1 shows a toy example of network configuration on the left-hand side and the corresponding attack graph on the right-hand side.

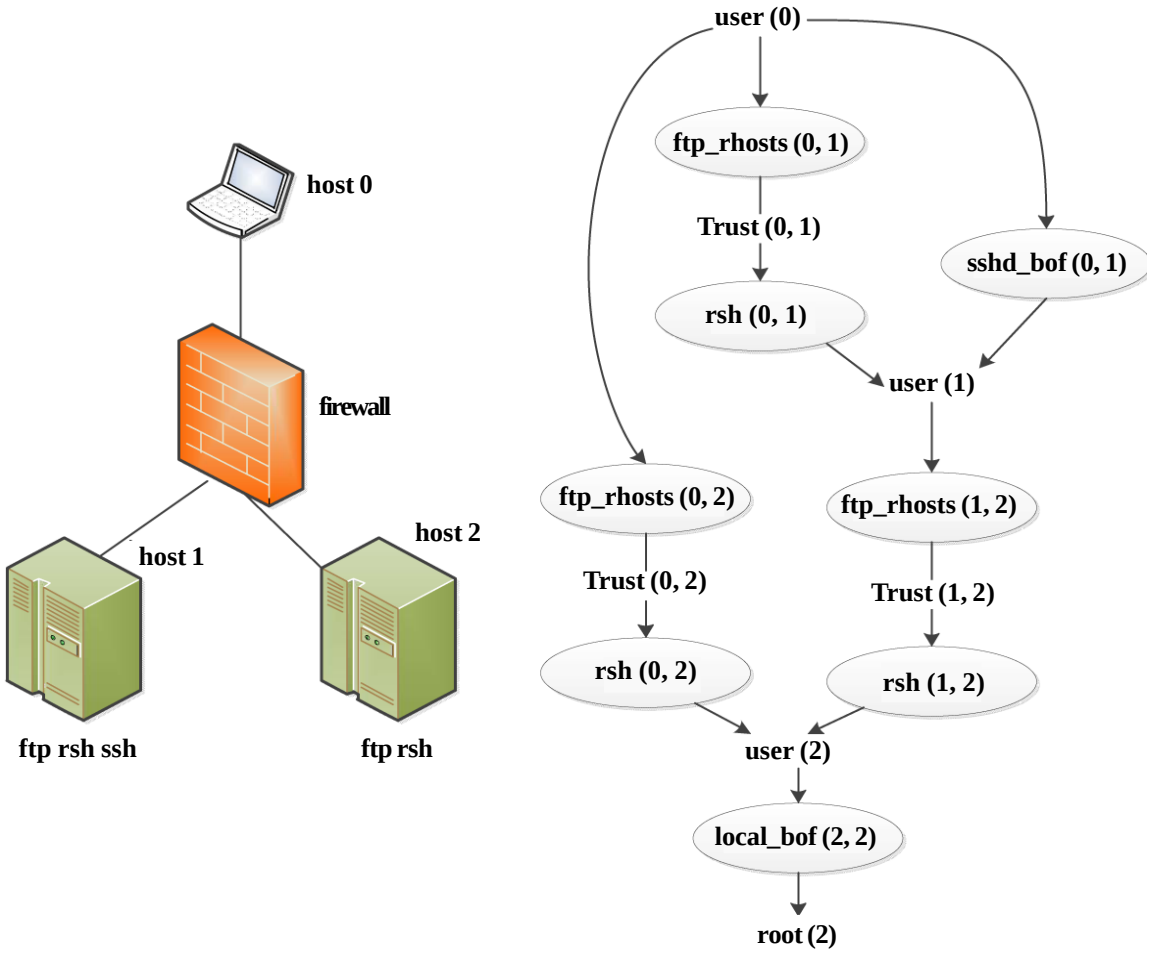


Figure 1. Network Configuration and Attack Graph

Figure 1 depicts a simple scenario where a file server (host 1) offers the File Transfer Protocol (ftp), secure shell (ssh), and remote shell (rsh) services; a database server (host 2) offers ftp and rsh services. The firewall only allows ftp, ssh, and rsh traffic from a user workstation (host 0) to both servers. In the attack graph, exploits of vulnerabilities are depicted as predicates in ovals and conditions as predicates in clear texts. The two numbers inside parentheses denote the source and destination host, respectively. The attack graph represents three self-explanatory sequences of attacks (attack paths). For example, the right path is: *sshd_bof* (0, 1) → *ftp_rhosts* (1, 2) → *rsh* (1, 2) → *local_bof* (2).

3.3 Attack tree

Attack models are used frequently in the context of computer networks and power control systems. Traditionally attack trees have been the most common type of model for representing known cyber-attacks [17, 18]. In an attack tree, the root of the tree represents the ultimate goal while the branches show all possible sequences of action steps towards the goal. An attacker might be imagined proceeding up the tree, reaching a new sub goal at each node. Thus, the modeling approach implemented in an attack tree visualizes an attack as a hierarchy of sub goals leading to the ultimate goal. The basic attack tree may be made more complicated in various ways, for example, nodes might have associated values or logical "*and/or*" conditions [19].

Ten et al. proposed to use attack trees for modeling cyber intrusions in existing power control systems [20]. Attack trees were shown to offer a systematic way to identify vulnerabilities of SCADA systems [11, 12 and 14] and quantify different vulnerability scenarios. McLaughlin, Podkuiko, and McDaniel presented an attack tree to illustrate potential ways to commit energy theft in the smart grid [21]. Their attack tree shows three classes of attacks, depending on how demand data is tampered with.

Attack trees are a popular modeling approach because they are good at describing an attack in an intuitive visual way; show all

attack paths within a broad picture; and can lead to useful mathematical analyses (e.g., risk assessment, vulnerability analysis) if nodes are assigned values. On the other hand, attack trees are somewhat limited in their view of attacks only proceeding in sequential steps. Also, they tend to focus on vulnerabilities, a single goal, and a single attacker. In this paper, we are concerned with Colored Petri nets because they do not have the limitations of attack trees.

4. Petri Nets and Colored Petri Nets

In this section, we introduce and define Petri Nets and Colored Petri Nets.

4.1 Petri nets

Petri Nets (PNs) was created in August 1939 by Carl Adam Petri for the purpose of describing chemical processes. A PN is a mathematical modeling language. It consists of places, transitions, and arcs that connect them. Input arcs connect places with transitions. Output arcs start at a transition and end at a place. Places can have tokens; the current state of the modeled system is given by the number and type of tokens in every place.

PN's model activities using places and transitions. A place represents a system in a time n, a transition connects two or more places. Transitions are only allowed to fire if they are enabled, which means that all the preconditions for the activity must be fulfilled. The preconditions ensure that there are enough tokens available in the input places. When the transition fires, it removes tokens from its input places and adds some into the output places. The number of tokens removed or added depends on the cardinality of each arc. The interactive firing of transitions in subsequent markings is called the token game.

PN's are good enough for describing and studying systems that are characterized as being concurrent, asynchronous, distributed, parallel, nondeterministic, and stochastic. Since PN's are a graphical tool, they can be used as a visual-communication aid similar to flow charts, block diagrams, or networks. Moreover, tokens are used in these nets to simulate the dynamic and concurrent activities of systems. In a PN is possible to set up state equations, algebraic equations, and other mathematical models governing the behavior of systems.

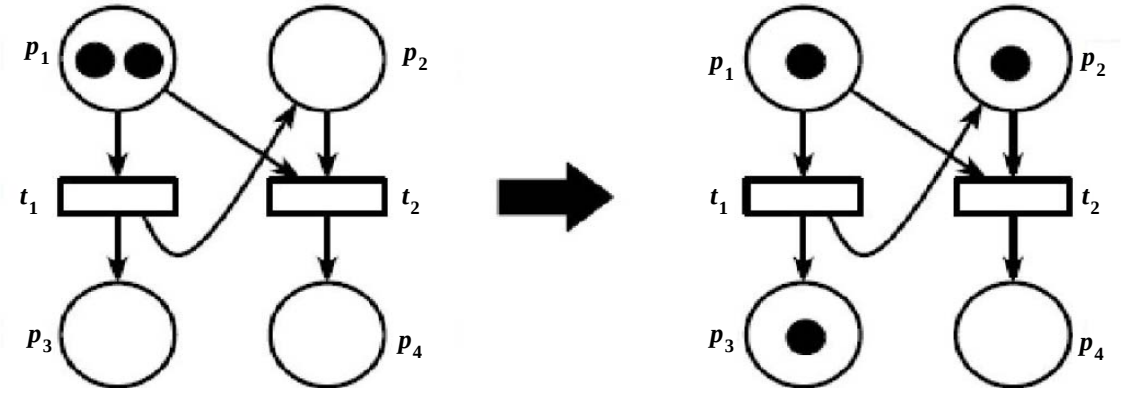


Figure 2. A Petri net before the firing of transition t_1

4.2 Colored petri nets

CoPNets were introduced as a full-fledged language for the design, specification, simulation, validation and implementation of large software systems. CoPNets combine the strength of Petri nets with the strength of programming languages and are widely employed in both academical and industrial areas for software system design, implementation, simulation and validation. CoPNets have a series of good features that make it suitable for modeling and analyzing complex systems [9, 10].

Each CoPNet has a set of declarations, which we position by convention in a box with dashed lines. The declarations introduce a number of color sets, functions, operations, variables and constants, which can be used in the net inscriptions of the corresponding CoPNet, particularly in the guards, arc expressions and initialization expressions. The declarations of a CoPNet can be made in many different languages, e.g., by means of standard mathematical notation or by means of many sorted sigma algebras. Each color set declaration introduces a new color set, whose elements are called colors. Every color set declaration

implicitly declares a set of constants (the colors of the color set). Moreover, the color set declaration can implicitly declare some standard functions and operations which can be used on the colors of the color set. A declared color set can be used [1]:

• In the declaration of another color sets.

• In the declaration of variables (having the color set as type).

• In the declaration of functions, operations and constants (e.g., a function may map from one color set into another color set).

• In the color set inscription of a place (indicating that all tokens on the place must have token colors which belong to the color set).

Definition 1 (CoPNet)
A CoPNet can be represented as a tuple CPN = $(\Sigma, P, T, A, N, C, G, E, M_0)$, where:

• Σ is a finite set of color sets, P, T and A are finite sets of places, transitions and arcs, respectively. $P \cap T = P \cap A = T \cap A = \phi$, and $A \subseteq P \times T \cup T \times P$. There are two types of arcs for a transition: incoming arc and outgoing arc. When a transition takes place, incoming arcs indicate that the input places shall remove specified number of tokens, while outgoing arcs mean that the output places should add the specified number of tokens. The exact number is determined by the arc expression, defined by the expression function E.

• N is a *node* function $N: A \to P \times T \cup T \times P$ and it specifies the source and destination of an arc.

• C is a *color* function, and $C : P \to \Sigma$. $C(p)$ specifies the set of allowed colors for any token of place p. A token element is a pair (p, c), where $p \in P$ and $c \in C(p)$. The set of all token elements is denoted as $TE = \{(p, c) / p \in P \wedge c \in C(p)\}$.

• G is a guard function, mapping each transition t to a boolean expression $G(t)$. Let IB stand for boolean type, which contains the elements {true, false}. Let *Type* (v) denote the type of the variable v, and *Type* (*expr*) stands for the type of the expression *expr*. *Var* (*expr*) denotes the set of variables in expression *expr*.

So : $\forall t \in T$: *Type* ($G(t)$) = IB $\wedge$ *Type* (*Var* ($G(t)$)) $\subseteq \Sigma$.

• E is an arc expression function, mapping each arc into an expression with *type* $C(p)$,where p is the place of the given arc. That is

$\forall a \in A$: *Type* ($E(a)$) = $C(p) \wedge$ *Type* (*Var* ($E(a)$)) $\subseteq \Sigma$.

• M_0 is the initial marking of CoPNet, and $M_0 \in (TE)$.

Definition 2 (Firing Rule)
A transition $t \in T$ is firable (or enabled) at a marking M if and only if : $\forall p \in P$: ($M(p) \geq W(p, t)$), where $W(p, t)$ is the weight of the arc to transition t from its input place p. Firing (or executing) transition t results in changing marking M to a reachable marking M', where : $\forall p \in P$: ($M'(p) = M(p) - W(p, t) + W(t, p)$) .

Definition 3 (Incidence Matrix)
The preincidence matrix *PRE* of a net N is $|P| \times |T|$ matrix row p and column t is the weight $W(p, t)$ of the arc from place p to transition t. The postincidence matrix *POST* of N is a $|P| \times |T|$ matrix whose element is the weight $W(t, p)$ of the arc from transition t to place p.

V = *POST-PRE* is called the incidence matrix of N.

To illustrate some of the basic concepts of a CoPNet consider the CoPNet in Figure 3.

There are three places (Sensitive Data Accessed, Communication Requested, and Leak) and one transition (the transition guard expression compares process IDs). The color set of the place "*Sensitive Data Accessed*" is positive integers. The color sets of the places "*Communication Requested*" and "*Leak*" are products of sets of positive integers and strings.

The distribution of tokens over the places is called marking. For the marking presented in Figure 3 the transition is enabled by tokens with PID = 3. This means that transition may occur and move the CPN into the next marking by removing tokens with PID = 3 from places "*Sensitive Data Accessed* " and "*Communication Requested* " followed by placing of a token into the "*Leak*"

place (see Figure 4). In the resulting marking the CPN has tokens in places "*Communication Requested* " and "*Leak*".

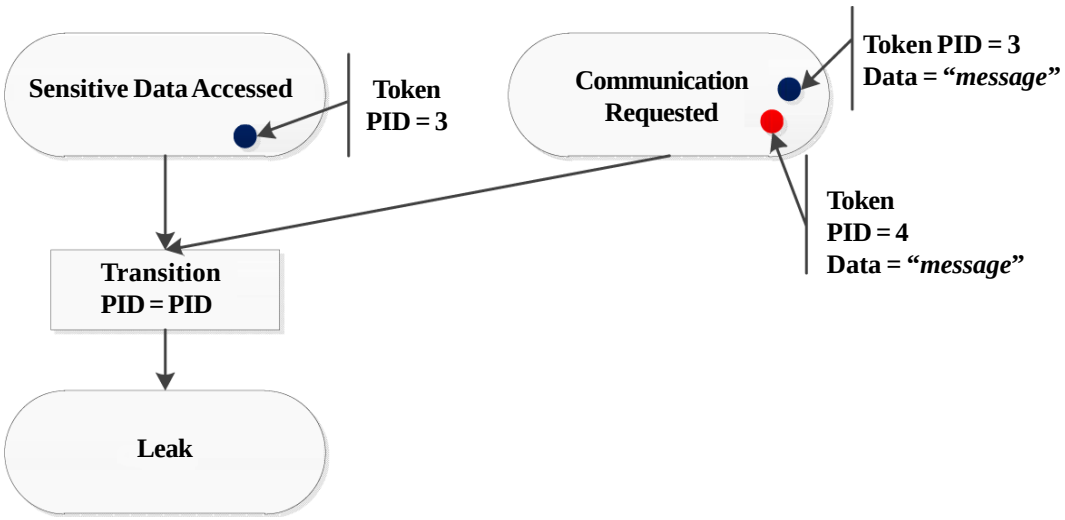


Figure 3. Sensitive data encryption decision making (step 1)

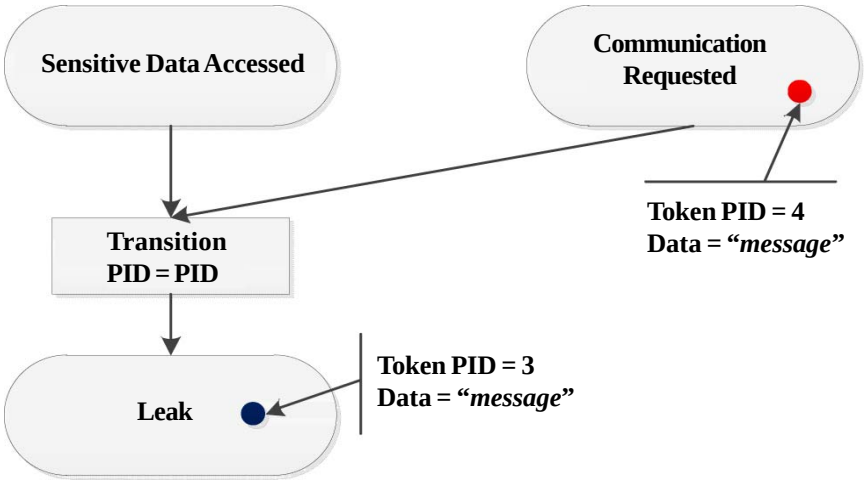


Figure 4. Sensitive data encryption decision making (step 2)

In the resulting state the token residing in the "*Leak*" place reflects the following facts:

• The program has accessed sensitive data

• The same program requested communication

This constitutes a complex event which by itself means more than a mere combination of events it consists of. This event indicates the possibility of sensitive data leak from the system.

5. Building CoPNet from Attack Tree

In this section we describe the transformation from ATs for intrusions to CoPNet approach for intrusion detection systems.

Although CoPNet is a powerful modeling technique, attacks can also be modeled by it. The definition of colored petri net is

addressed in this section firstly. Then the mapping from AT into CoPNet based attack model is analyzed. Some other extended features of this model are also expressed in this section.

The CoPNet based attack model can be defined from AT to reduce the cost of modeling. It is because that some attack models have been built with ATs [2, 3]. To build a CoPNet based model from an AT, the mapping rules between them should be determined. The root node of AT is the result of an attack, and the leaf nodes are actions attacker exploiting to break into system. It is clearly that the root node of an AT can map to transitions. The relationship among nodes could be regarded as the arc of CoPNet. The node value in ATs is expressed as arc expression of CoPNet. And the logics of ATs can map to event relationship of CoPNet.

5.1 Root node mapping

In AT, root node is the goal and result of attack. In CoPNet attack models, the root node can map to place: node maps to a place, node inputs map to arcs of place. This kind of place is called root place. The OR gate and AND gate will map to the event relationship of CoPNet. Their maps are shown in Figure 5. The node with OR gate maps to event's conflict relation of CoPNet. This means only when one event occurs, will the attack take a place. The node with AND gate maps to event's sequential relation implies that only when all events occur, then the attack will take place.

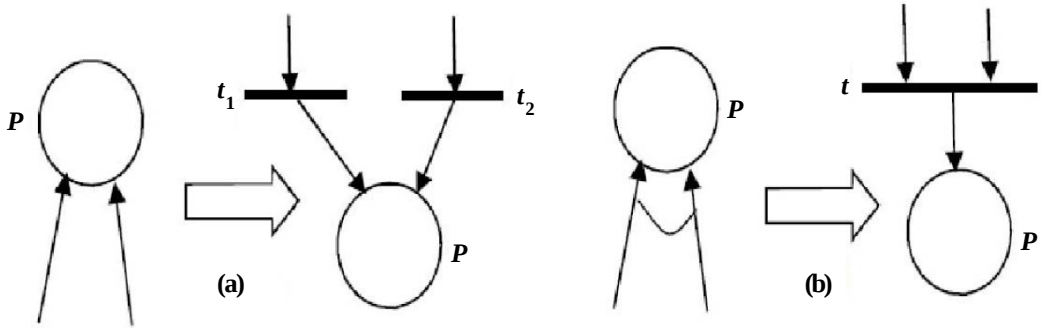


Figure 5. Root nodes and their mapping in CoPNet. (a) is OR gate of root node, and (b) is AND gate of root node

5.2 Leaf nodes mapping

Leaf nodes in ATs are attacker's actions breaking into the victim's system. It is clearly that leaf nodes can map to the transition of CoPNet. But in an AT, all leaf nodes are connected directly. So it is difficult to do straightforward maps. Some intermediate states must be defined so that the mapping can be performed. Ruis's analysis of intrusion divided attacks into seven stages: Reconnaissance, Vulnerability Identification, Penetration, Control, Embedding, Data Extraction & Modification, and Attack Relay [4]. Each stage can also be divided into some or several sub-stages. So we can model attacks' stages and sub-stages as intermediate states when translating ATs into CoPNet based attack models. Figure 6 shows how to deal with such translation. These newly added places (including the places added during translation of intermediate nodes) are called Added Place. In Figure 6, the value of place p can be derived from function $f(t)$, where $t \in T$, and $f(t) \in \Sigma$. And the output arc of place p is the input arc of next transition. In Figure 4, the *IP* place in an Initialization Place whose means depend on the transition.

5.3 Intermediate Nodes Mapping

Intermediate nodes of ATs are sub-actions or sub-goals of attackers. It is more difficult to translate these nodes into CoPNet models because intermediate nodes have not only input arc (s) and output arc (s), OR and AND gate logics, but also the same problems confronted in leaf node translating. Mapping rules of leaf nodes are listed as follow:

The intermediate node itself maps to transition, *t*, of CoPNet.

Input arc of an intermediate node maps the input arc of *t*, OR and AND gates are translated to confliction relation and sequence relation respectively.

Intermediate place is added in the same way as the translations of leaf nodes.

By using above rules, the intermediate node can be mapped into CoPNet attack model. Figure 7. Shows the mapping relations of

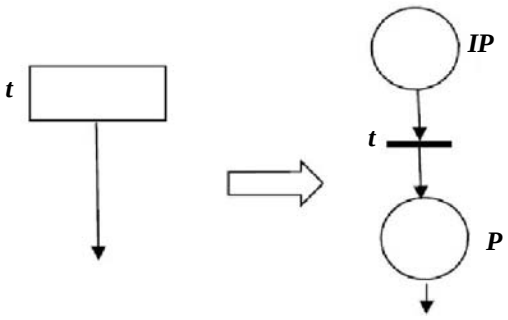


Figure 6. Leaf node and its mapping in CoPNet

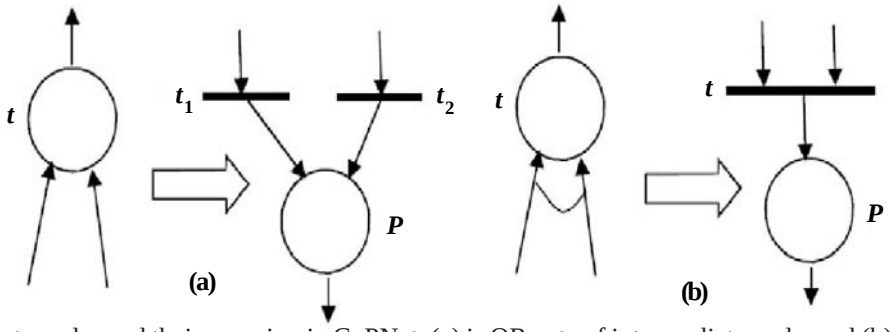


Figure 7. Intermediate nodes and their mapping in CoPNet. (a) is OR gate of intermediate node, and (b) is AND gate of intermediate node

a root node and a leaf node are mostly similar, other than that the latter has an output arc. But essentially, they are different from each other. In root node translation, the node itself maps to a transition and a place is added to connect newly added place with the corresponding transition.

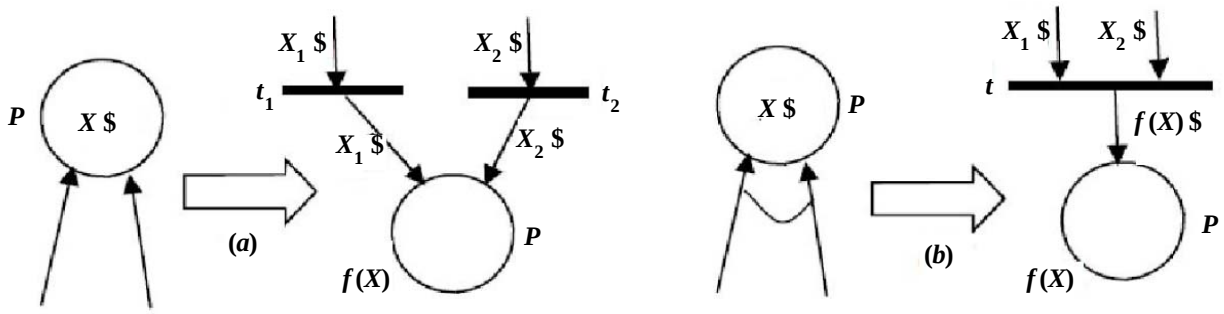
5.4 Temporal Logic Mapping

As to building template CoPNet from AT, an important issue is how to deal with temporal sequence of attack. From above discussing, we know that an attack comprises many stages and sub-stages that all have temporal logical relations. One occurring sequence of many stages and sub-stages means an intrusion while all occurring sequences comprise the AT. Even relations of CoPNet can depict temporal logics in an AT. Event relations of CoPNet can depict temporal logics in an AT. In fact, only sequence relation and conflict relation are used in CoPNet attack models. Although intermediate modes have multiple out arc, concurrence relation may also be used. In this paper, concurrence relations are note used.

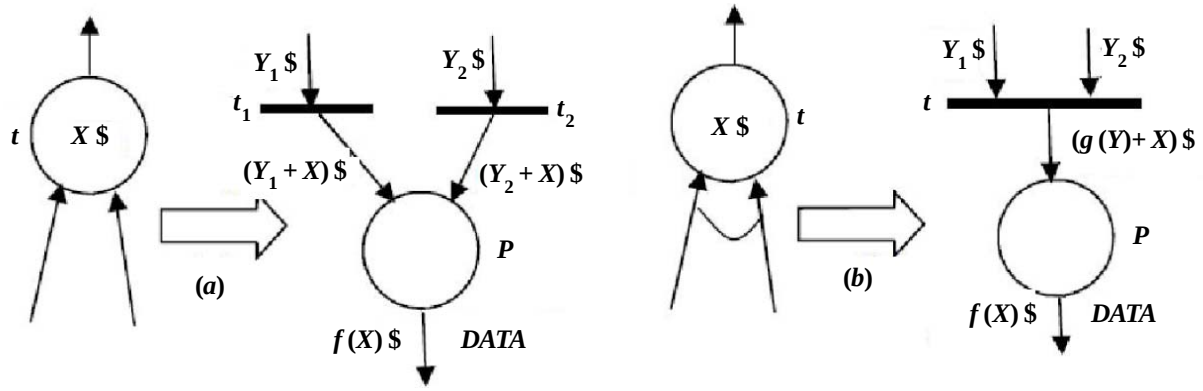
5.5 Node Value Mapping

The node value in AT is used to perform risk assessment. This special feature extends its application scope and usage for quantifying intrusions. In CoPNet, there is no node value function, but it can be expressed by color value of place of CoPNet. During the translation of CoPNet model, after transition is fired, some value will be added to arc expression of this transition *t*, and a color function maps each place, *p*, to a type *C* (*p*) that expresses the node value. So each token must have a data value to evaluate risk.

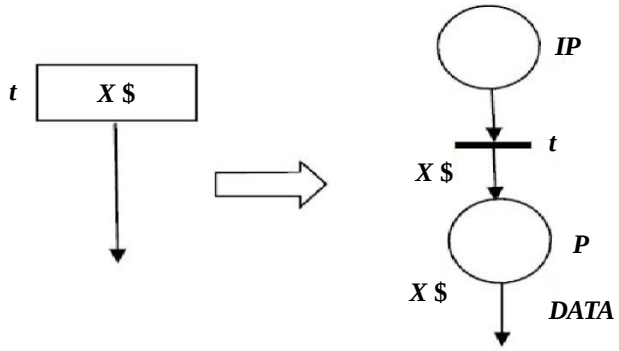
It is easy to translate the weighted leaf node: node value maps to output expression of newly added place is also evaluated to this value. For intermediate node with value, if the node has OR gate, the value should be mapped to value of output arc of transition by mapping function $f(X) = g(Y) + X$, where $g(Y) = \{y / y = Y_1 \wedge y \neq Y_2 \text{ or } y = Y_2 \wedge y \neq Y_1\}$; if the node has AND gate, node value will be mapped to output arc expression whose value is evaluated by $f(X) = g(X) + X$, where $g(Y) = Y_1 + Y_2$. Translations of leaf nodes and intermediate nodes are more difficult than that of root node. As to root node with node value, if root node has OR gate, it can be calculated from the input arc expression of transition added during translation by function $g(X) = \{x/x = X_1 \wedge x \neq X_2 \text{ or } x = X_2 \wedge x \neq X_1\}$; if root node has an AND gate, node value (\$ denotes) $f(X)$ includes two parts, $X_1\$$ and $X_2\$$ where $f(X) = X_1\$ + X_2\$$.



(1) Root nodes with node value. (a) is OR gate of root node, and (b) is AND gate of root node



(2) Intermediate nodes with node value. (a) is OR gate of intermediate node, and (b) is AND gate of intermediate node



(3) Leaf node with node value

Figure 8. Node with node value and their mapping

6. Extended CoPNet Model for Intrusion and Response

AT only depicts the process of an attack. No mechanism in AT is provided to allow active response and defense, partially due to limits of tree model. But CoPNet based attack modeling can give administrators such means to control the hacker's action or carry out some effective response. Based on the definition of CoPNet, transition can fire only when all its bindings occurs. So we can model the defense and response actions as follows: for each transition, an input arc is added to allow control, and an output arc is added to allow response. Additionally, if there are many control and response actions, many arcs can also be added. But readers should be aware that this model is not derived from AT, but extends directly from CoPNet attack model.

7. Case Studies

In this section, we describe the attack selection process for a case study process control network for a power grid, and we use this case study to illustrate the usages of CoPNet based attack modeling approach. We have chosen BGP (Border Gateway Protocol), SCADA (Supervisory control and data acquisition) systems and malicious insider attacks (MIA) as our three case studies.

7.1 Border Gateway Protocol Networks

An example scenario for a BGP attack is shown in Figure 9. An attacker prevents two peers from exchanging routing information by repeatedly causing a BGP session in Established state to reset. The BGP session can be reset by injecting a spoofed TCP (Transmission Control Protocol) or BGP message into the router message stream. Building a valid TCP/BGP packet requires a valid TCP sequence number (obtained by TCP sequence number prediction). During the initial stages of a TCP sequence number attack, a spoofed packet from an attacker is usually followed by the original packet from the authentic source. Spoofed TCP message with RST flag set will cause a connection to reset. Spoofed BGP messages (OPEN, NOTIFICATION or KEEPALIVE messages) received by the BGP speaker in the Connect or Active states will cause the router to reset resulting in a denial of service. The BGP speaker can also be compromised by gaining physical or logical (hijacking a router management session) access to the router.

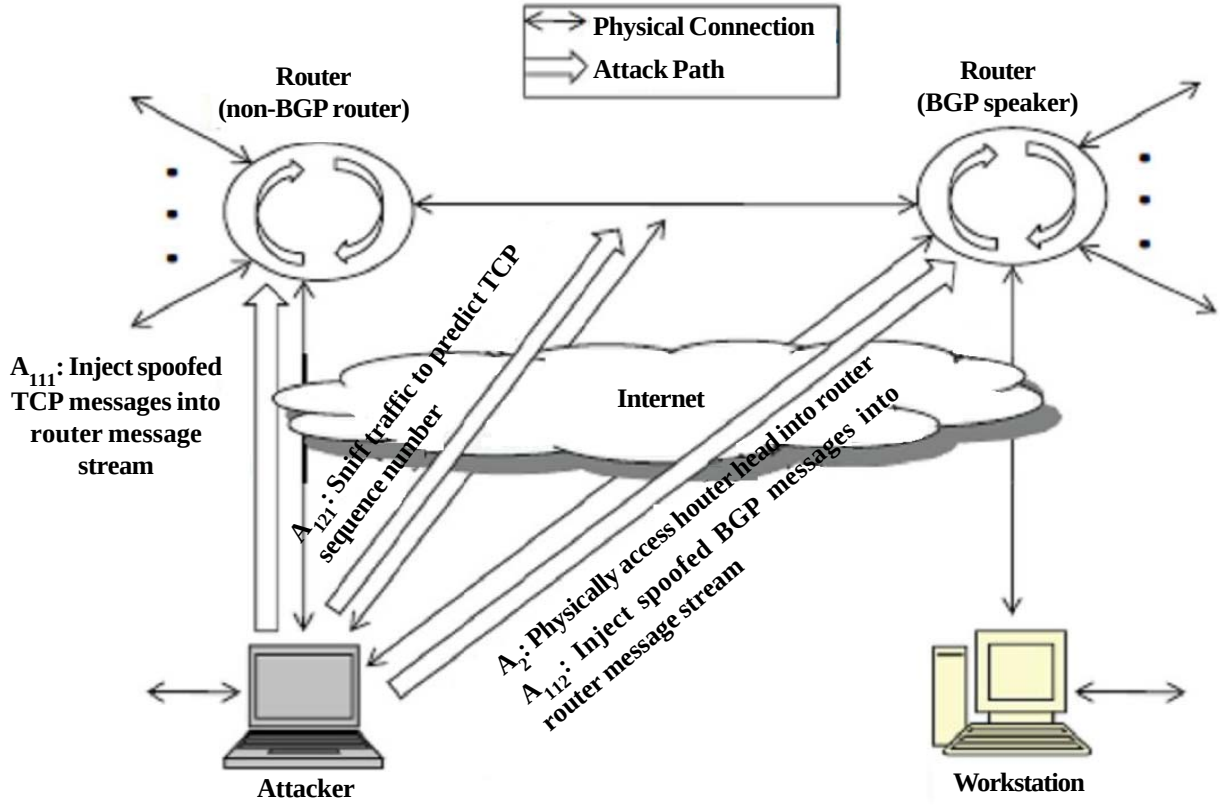


Figure 9. Example of attack for resetting a BGP session

Figure 10 shows an example of AT for BGP attack ("*resetting a BGP session*" [23]). As shown in Figure 8, the tree with different costs assigned to the leaf nodes. The "$" is the cost of attack. Like Boolean node values, these processes can propagate up the tree as well. OR nodes have the value of their cheapest child; AND nodes have the value of the sum of their children. Obviously, the costs in Figure 8 have propagated up the tree, the cheapest attack has been highlighted and so the tree in this figure is called minimum cost AT. Hence, it is difficult to depict all cost features of an attack can be attained in one tree.

Figure 11 shows CoPNet based attack model of BGP networks. The $f_1(x)$ is a function whose value depends on the fired

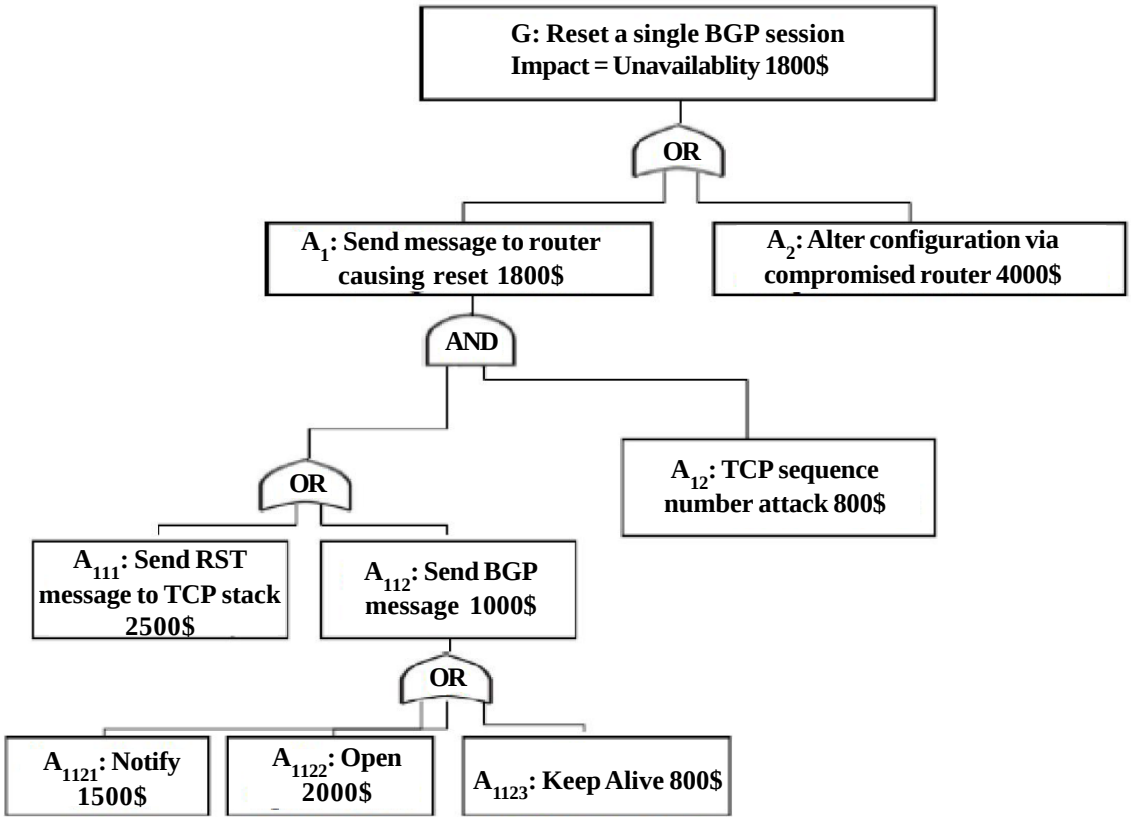


Figure 10. An AT for attacks involving resetting a BGP session

whose value depends on the fired transitions: t_4, t_5 and t_6. ML language is adopted to define functions and variables. In CoPNet based attack modeling, all cost features are described in one model. If different values are given, different cost model can be derived from the same CoPNet model. Additionally, from CoPNet based model, the attack process and states of victim can be clearly attained through state space analysis of CoPNet. If some time-related features are added to such CoPNet model, it is also easy to test and verify the temporal logic and its performance of CoPNet attack model.

7.2 Supervisory Control and Data Acquisition Networks

Figure 12 shows a sample 3-bus power grid and its SCADA network [27], which is responsible for monitoring and controlling the underlying power system. There are a total of three generators, any one of which is able to provide the power required by customers, i.e., load. To monitor the power system, each bus is attached to a sensor, i.e., a phasor measurement unit (PMU). The sensor sends voltage phasors (i.e., magnitudes and phase angles) of the bus and current phasors of transmission lines connected to that particular bus to SCADA. Moreover, to control power generation, having received sensory data, SCADA computes optimal generation set points for individual generators. As shown in Figure 12, SCADA consists of different components, among which there are constrained communications. First of all, given noisy sensory data, the state estimation server is responsible for estimating the state of the whole power system. A database stores these states and other information that might be used later by administrators or customers through the web server. The human machine interface (HMI) and security constrained optimal power flow (SCOPF) compute control commands using those estimated states. As demonstrated, a hot spare HMI is also active and connected as part of the network.

Figure 13 illustrates a sample brief network-level AT for the process control network described above. The top event is chosen to be "*SCADA is compromised*," and its children denote deficiencies in providing loads and report generation, which are two main goals of the supervisory network. For simplicity, leaf nodes here denote compromise of individual host systems, and are updated by local engines. As a case in point, *G*1, if set to 1, indicates that the controller device for the generator on bus 3 is compromised.

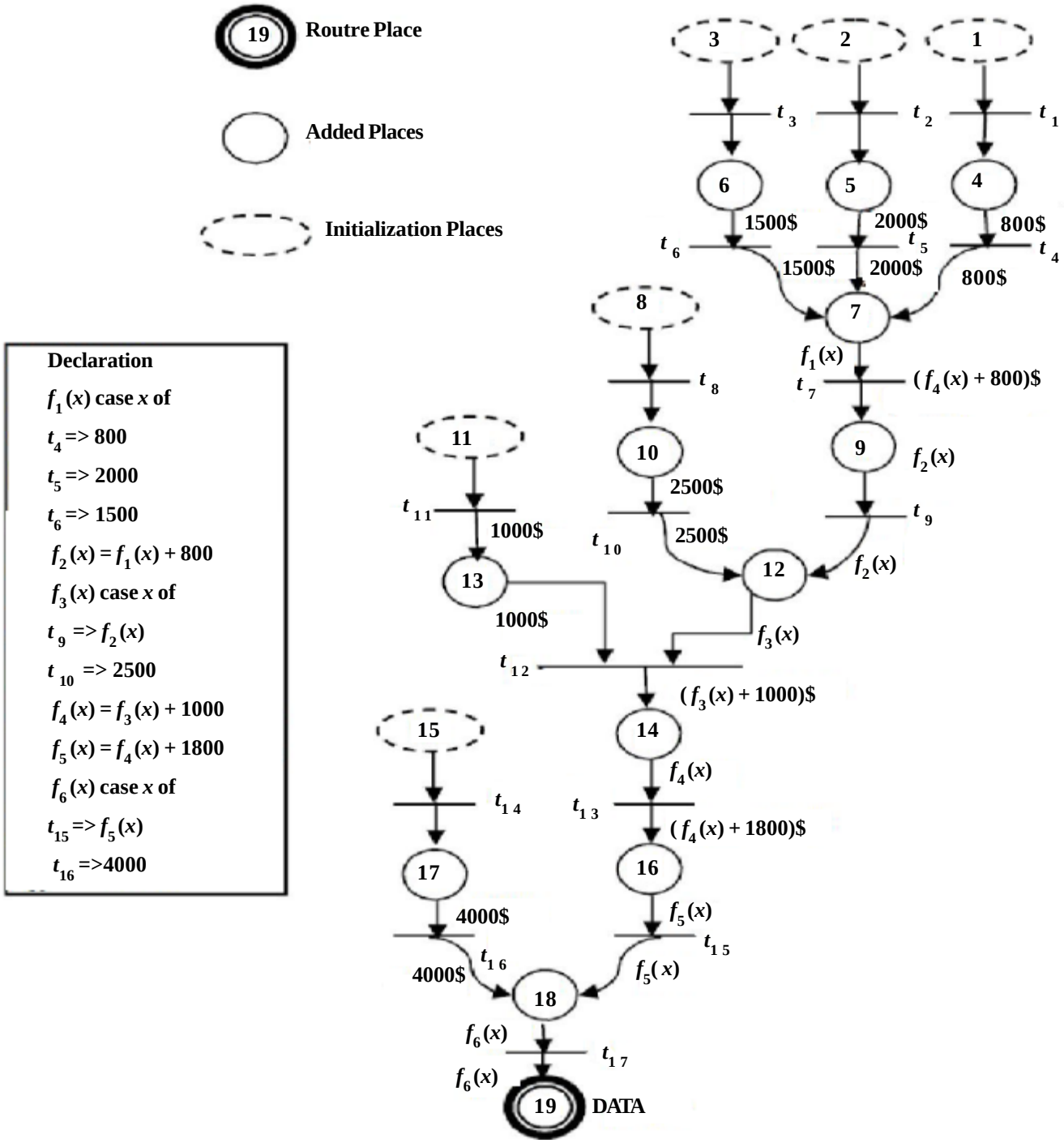


Figure 11. CoPNet based attack model of BGP session

As shown in Figure 13, the tree with different costs assigned to the leaf nodes. The "$" is the cost of attack. Like Boolean node values, these processes can propagate up the tree as well. OR nodes have the value of their cheapest child; AND nodes have the value of the sum of their children. Obviously, the costs in Figure 13 have propagated up the tree, the cheapest attack has been highlighted and so the tree in this figure is called minimum cost AT. Hence, it is difficult to depict all cost features of an attack can be attained in one tree.

Figure 14 shows CoPNet based attack model of SCADA networks. The f_1 (x) is a function whose value depends on the fired

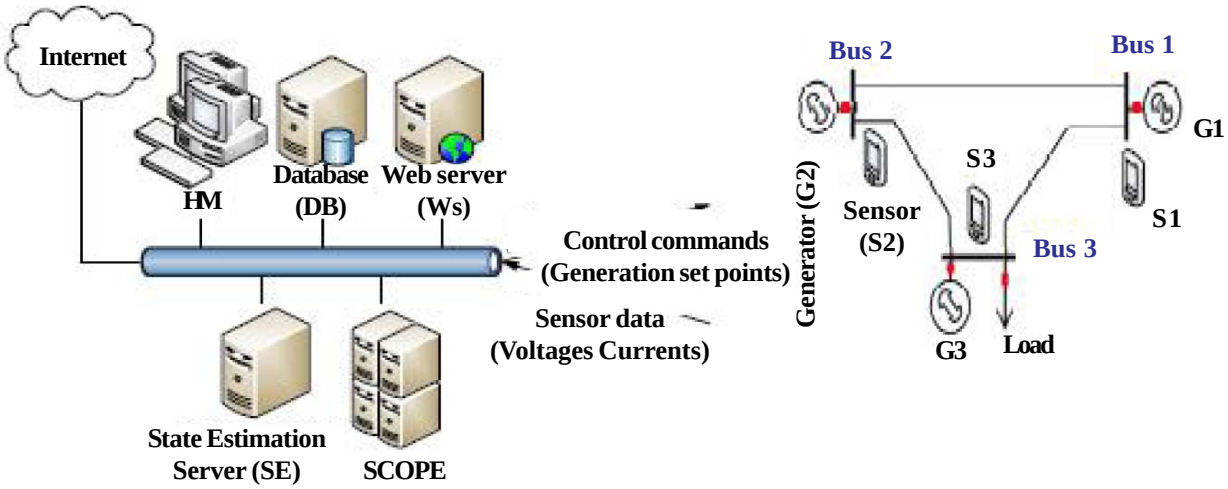


Figure 12. SCADA Networks

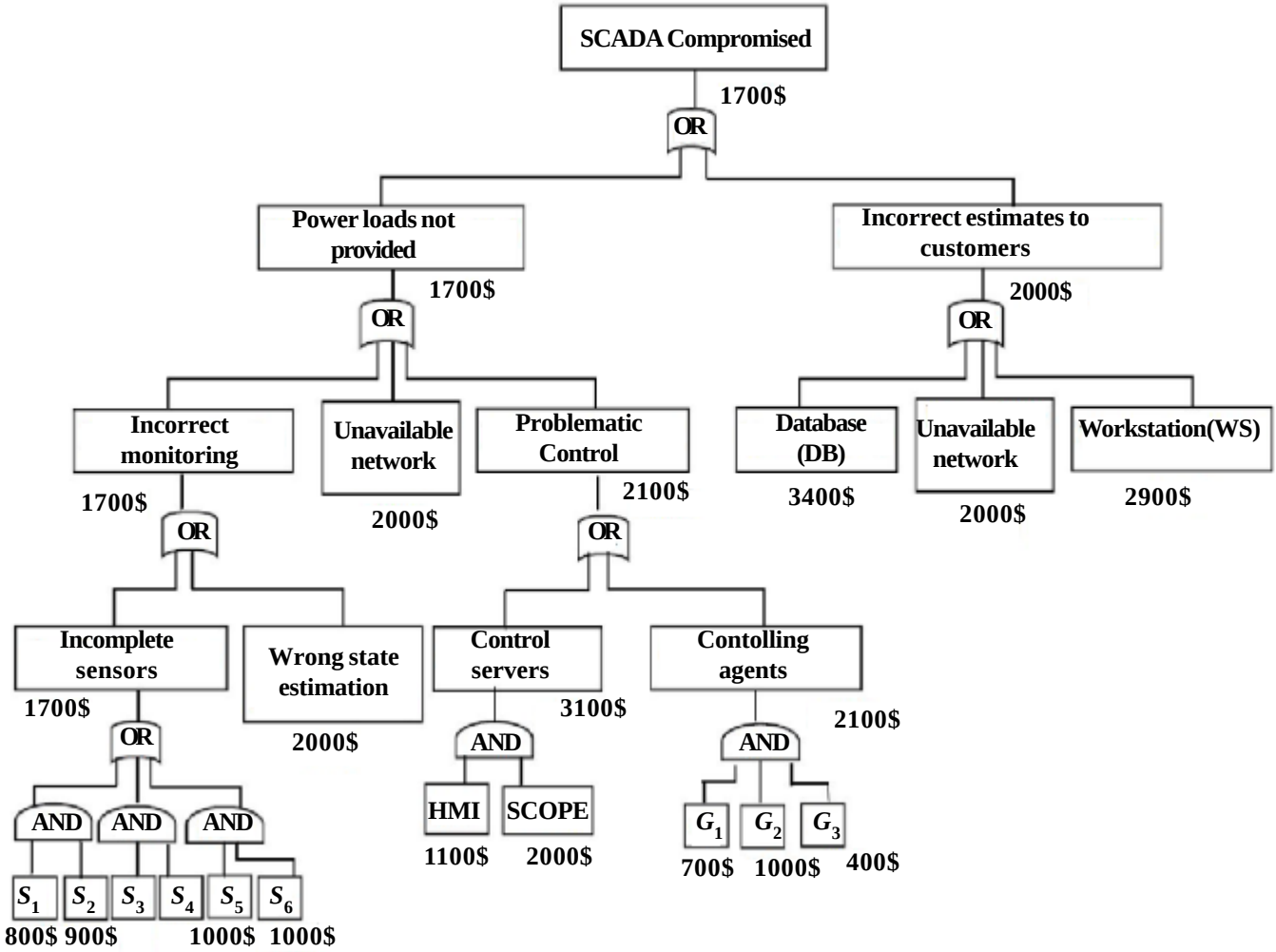


Figure 13. An AT for attacks involving resetting a SCADA session

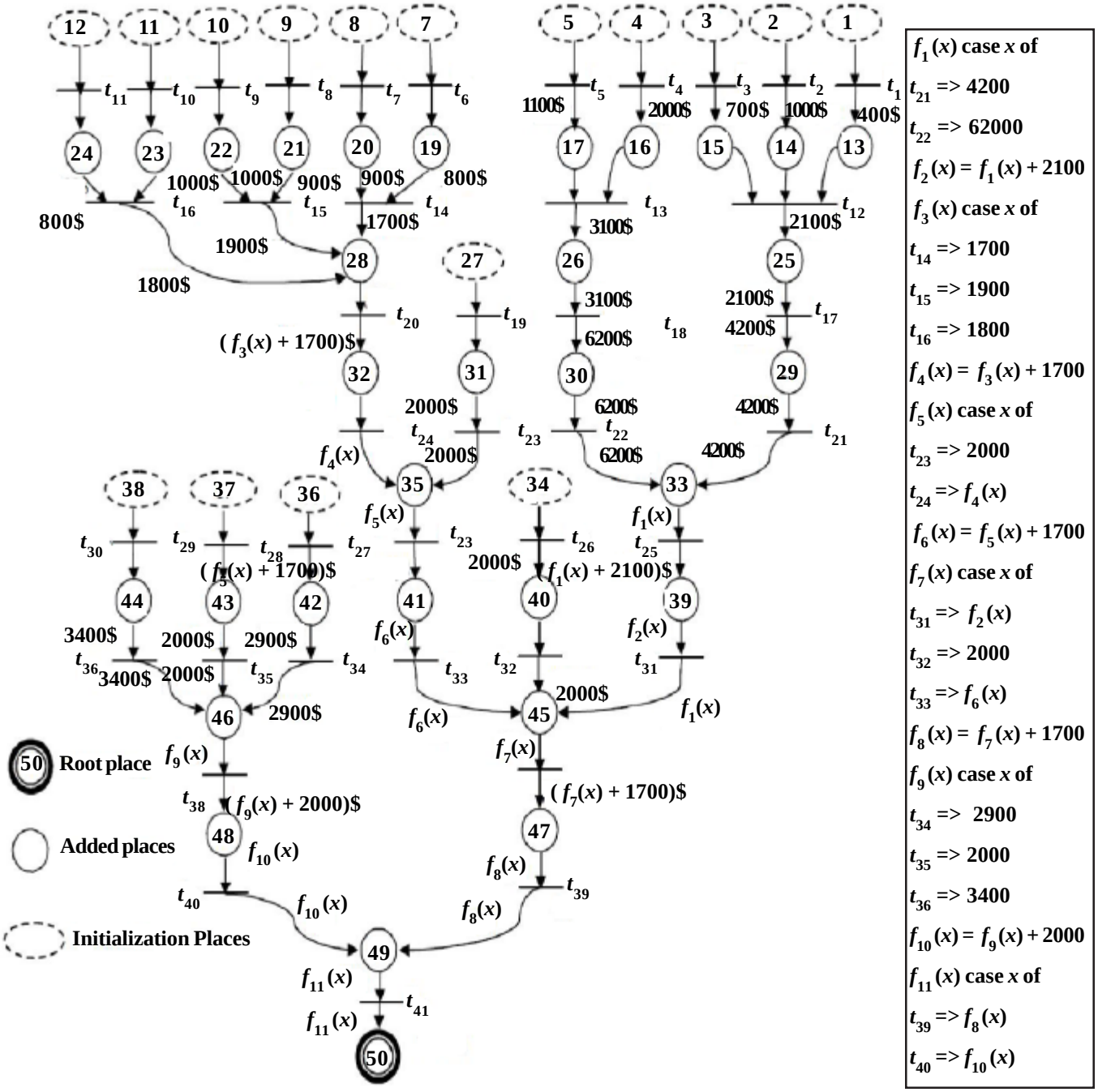


Figure 14. CoPNet attack model of SCADA session

transitions: t_{21} and t_{22}. ML language is adopted to define functions and variables. In CoPNet based attack modeling, all cost features are described in one model. If different values are given, different cost model can be derived from the same CoPNet model. Additionally, from CoPNet based model, the attack process and states of victim can be clearly attained through state space analysis of CoPNet. If some time-related features are added to such CoPNet model, it is also easy to test and verify the temporal logic and its performance of CoPNet attack model.

7.3 Malicious Insider Attacks

The basic structure of the AT for malicious insider attack (MIA) was proposed in [18]. Figure 15 illustrates a sample brief

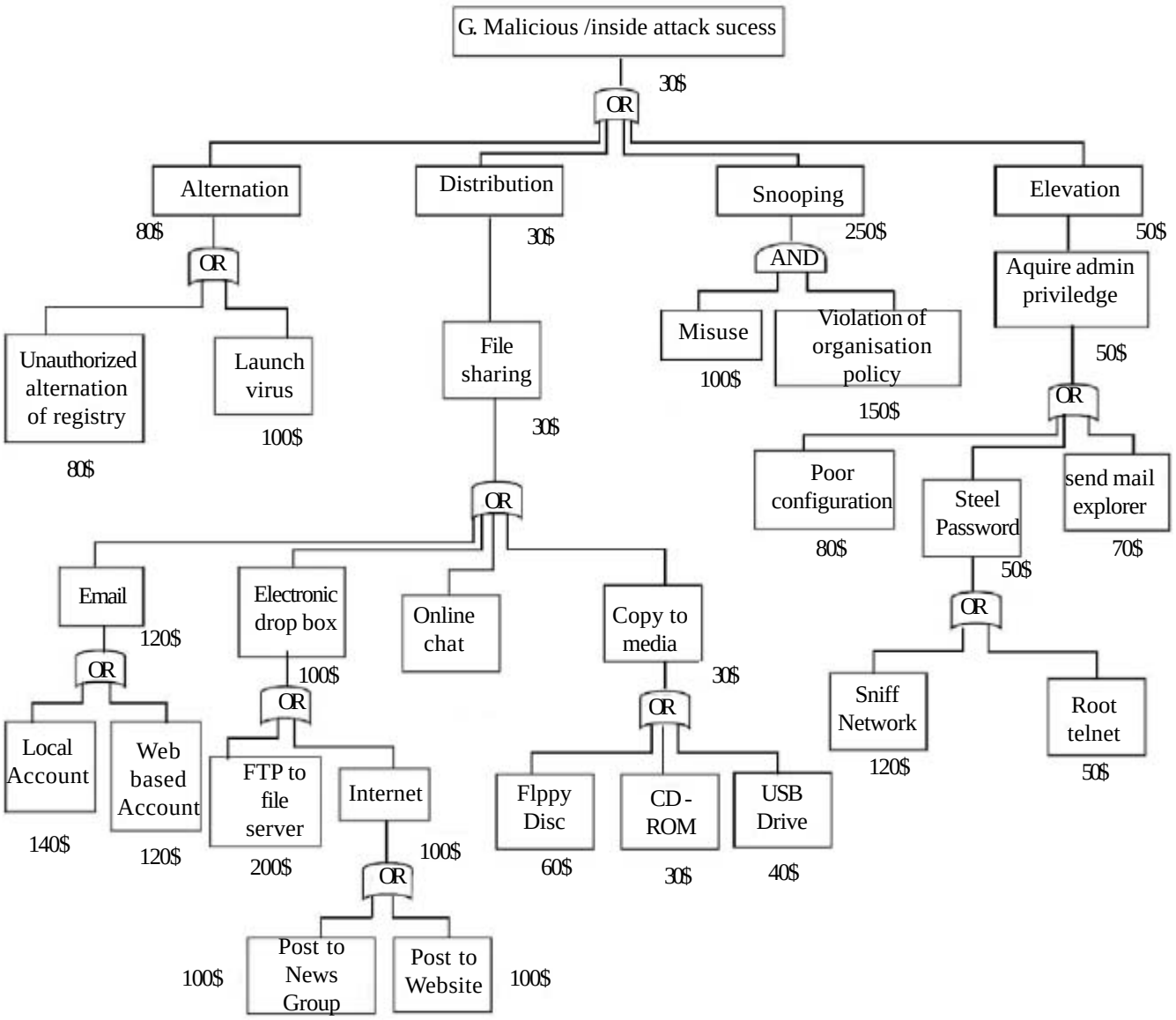


Figure 15. An AT for Malicious Insider Attack (MIA)

network-level AT for the process control network described above. The top event is chosen to be "*Malicious Insider attack success*". As shown in Figure 15, the tree with different costs assigned to the leaf nodes. The "$" is the cost of attack. Like Boolean node values, these processes can propagate up the tree as well. OR nodes have the value of their cheapest child; AND nodes have the value of the sum of their children. Obviously, the costs in Figure 15 have propagated up the tree, the cheapest attack has been highlighted and so the tree in this figure is called minimum cost AT. Hence, it is difficult to depict all cost features of an attack can be attained in one tree.

Figure 16 shows CoPNet based attack model of MI AT. The $f_1(x)$ is a function whose value depends on the fired transitions: t_3 and t_4. ML language is adopted to define functions and variables. In CoPNet based attack modeling, all cost features are described in one model. If different values are given, different cost model can be derived from the same CoPNet model. Additionally, from CoPNet based model, the attack process and states of victim can be clearly attained through state space analysis of CoPNet. If some time-related features are added to such CoPNet model, it is also easy to test and verify the temporal logic and its performance of CoPNet attack model.

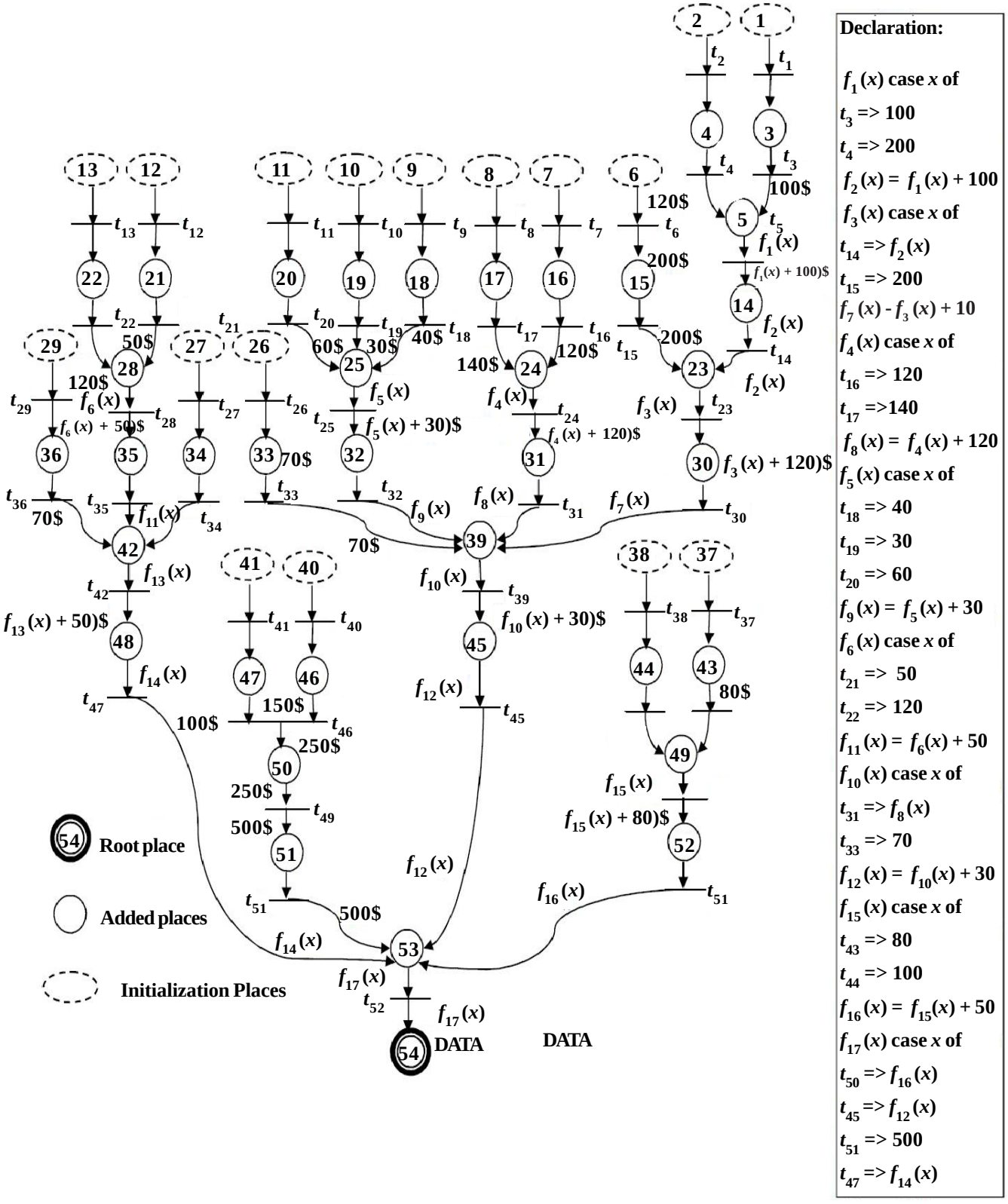


Figure 16. CoPNet based attack model of MI attacks

8. Conclusion and Future Work

In this work, we have presented CoPNet based attack modeling approach to model the attacks. The objective of our modeling approach is to provide more precise quantitative parameterization and advanced modeling capabilities compared to ATs. After other features are added to this model, it can be used to model the intrusion detection and response. Another important feature of this model is that intrusion can be quantified, so the most effective controlling actions can be determined. But the practical experiment shows the CoPNet based attack model has a more complicated form than the graph-like model, especially AT. So it is necessary to condense the CoPNet based attack model. This goal can be achieved by using the ML language. Afterward we will further explore some CoPNet place reducing methods to simplify CoPNet attack model. We have provided three case studies that illustrate the CoPNet approach (BGP attack, SCADA attack and malicious insider attack). We have showed that CoPNet based attack model has many unique characters which AT model has not. Simulation approach of CoPNet based attack model is our future work.

References

[1] Jensen, K. (1992). Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use. *Springer-Verlag*, Berlin, Germany/ Heidelberg, Germany/ London, UK/ ect., 1.

[2] Schneier, B. (1999). Attack Trees. *Dr. Dobb's Journal of Software Tools*, 24 (12) 21-29.

[3] Cunningham, W. (2002). The WikiWikiWeb [DB/OL]. http:// c2.com/cgi-bin/wiki.

[4] Ruiu, D. (1999). Cautionary tales: stealth coordinated attack how to [DB/OL]. http://www.nswc.navy.mil/ISSEC/CID/Stealth_coordi nated_Attack .html, July.

[5] Bugtraq. (2003). Vulnerability Database [DB/OL]. http:// www.securityfocus.com.

[6] Helmer, G., Wong, J., Slagell, M., et al. (2001). A Software Fault Tree Approach to Requirements Analysis of an Intrusion Detection System [C]. *In*: Proceeding of symposium on requirements engineering for information security. Center for education and research in information assurance and security, Perdue University, March.

[7] Phillips, C., Swiler, L P. (1998). A Graph-based System for Network-Vulnerability Analysis [C]. *In*: Proceeding of new security paradigms workshop, Charlottesville, VA, USA, p. 71-79.

[8] Helmer, G., Wong, J., Slagell, M., et al. (2007). Software Fault Tree and Colored Petri Net based specification and implementation of agent- based intrusion detection systems. *Int. J.Information and Computer Security*, 1 (1/2).

[9] Jensen, K. (1998). A brief introduction to Colored Petri Nets, *Workshop on the Applicability of Formal Models*, 2 June, Aarhus, Denmark, p. 55-58.

[10] Jensen, K. (1998). An introduction to the Theoretical Aspects of Colored Petri Nets, *Workshop on the Applicability of Formal Models*, Aarhus, Denmark.

[11] The center for SCADA security, The Center for SCADA Security, *Sandia National laboratory*. [online]. Available http://www.sandia.gov/scada/home.htm.

[12] Supervisory Control and Data Acquisition (SCADA) Systems. (2004). Technical Information Bulletin 04-1, *National Communication System*, Oct. [online]. Available: http://www.ncs.gov/library/tech_bulletins/2004/tib_04- 1.pdf.

[13] Understanding SCADA system security vulnerabilities. (2005). *Symantec White Paper*.

[14] Brown, T. (2005). Security in SCADA systems: How to handle the growing menace to process automation, *IEE Comp. and Control Eng*., 16 (3), Jun./Jul. p.42-47.

[15] Rehman, R. (2003). Intrusion Detection Systems with Snort. *Prentice-Hall*.

[16] Lippmann, R., Ingols, K. (2005). An annotated review of past papers on attack graphs. Technical report, *MIT Lincoln Laboratory,* March.

[17] Mauw, S., Oostdijk, M. (2005). Foundations of attack trees, *In*: Proc. 8th Annu. Int. Conf. Inf. Security Cryptol. (ICISC), Seoul, Korea, Dec. p. 186–198.

[18] Khand, P. (2009). System level security modeling using attack trees, *In*: Proc. 2nd Int. Conf. Comput., Control, Commun. (ICA), Karachi, Pakistan, Feb. p. 1–6.

[19] Schneider, K., Liu, C. -C., Paul, J.-P. (2006). Assessment of interactions between power and telecommunications infrastructures, *IEEE Trans. Power Sys.*, 21, p. 1123–1130, Aug.

[20] Ten, C.-W., Liu, C.-C., Govindarasu, M. (2007). Vulnerability assessment of cybersecurity for scada systems using attack trees, *in IEEE Power Eng. Soc. Gen. Meet.*, Tampa, FL, Jun. p. 1–6.

[21] McLaughlin, S., Podkuiko, D., McDaniel, P. (2009). Energy theft in the advanced metering infrastructure, *In*: Proc. 4th Int. Workshop Crit. Inf. Infrastruct. Security (CRITIS 2009), Bonn, Germany, Sep. p. 176–187.

[22] El Bouchti, A., Haqiq, A. (2012). Performance Modeling of Attack Countermeasure Using Colored Petri Nets, *International Symposium on Security and Safety of Complex Systems*, Agadir, Morocco, May 25 - 26.

[23] Convery, S., Cook, D., Franz, M. (2002). An Attack Tree for the Border Gateway Protocol, *Cisco Internet draft*.

[24] El Bouchti, A., Haqiq, A. (2012). Cyber-Attack for BGP using CoPNet Model, INTECH'12, Casablanca, Morocco, September 18-20.

[25] El Bouchti, A., Haqiq, A. (2012). Modeling Cyber-Attack for SCADA System using CoPNet Approach, ICCS'12, Agadir, Morocco, November 5-6.

[26] El Bouchti, A., Haqiq, A. (2013). Malicious Insider Attacks Based Colored Petri Nets Approach, *International Journal of Engineering and Technology Sciences (IJETS)*, Agadir, 4, p. 177-191.

[27] Baker GH, Berg A. (2002). Supervisory Control and Data Acquisition (SCADA) Systems. The Critical Infrastructure Protection Report 1.6 2002.