I. J. Computer Network and Information Security, 2018, 10, 1-10
Published Online October 2018 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2018.10.01

Verification of Web Content Integrity: Detection and Recovery Security Approach using Colored Petri Nets

Sherin Hijazi
King Abdullah II School for Information Technology the University of Jordan, Amman, Jordan
E-mail: sherinhijazi@yahoo.com

Amjad Hudaib
King Abdullah II School for Information Technology the University of Jordan, Amman, Jordan
E-mail: ahudaib@ju.edu.jo

Received: 24 June 2018; Accepted: 07 September 2018; Published: 08 October 2018

Abstract—This paper focuses on the design model verification processes to reduce modification cost after the software is delivered. We proposed a new design flow of web content integrity to protect web security by using colored petri nets simulation. The method covers the design process from the firewall stage to the recovery stage. In the proposed solution, the model verified the integrity of web content through detection tampering and recovery web content processes. Furthermore, the specification formally verifies the model checking technique by colored Petri nets formalism. Finally, the model is simulated by colored petri nets to insure the correct behavior of the designed web content integrity model.

Index Terms—Verification, Firewall, Web Content integrity, CPNs tool.

I. Introduction

Most companies rely on web sites to deliver services to their customers and interact with them. Web applications allow visitor access to the most critical resources of a web site, the web server and the database server. Web application still suffers from issues of security like web content integrity.

The web security protection requires the achievement of three concepts: integrity, availability and confidentiality of data. This paper focuses on the integrity of data that refers to the trustworthiness of information resources, thereby ensuring that only an authorized client can alter the data, as unauthorized access may result in the incorrect or malicious behavior of the web application. Therefore, it is important to confirm and verify all sages of software engineering in web security development to detect the web security vulnerabilities at each stage, instead of processing the security vulnerabilities at the implementation stage.

Verification methods are widely used to successfully address software security challenges and make software systems more trustworthy. Verification is the process of analyzing the properties, accuracy or validation of the system specification. Formal verification [29] is a method for proving the correctness of system specification when they are described in a mathematically rigorous framework. The behavior of the system must be modeled accurately for verification purposes. The formal verification method is for situation definitions and demonstrates its feasibility and efficiency.

A specification can be written formally in the form of control Petri nets, or semi-formally by unified modeling language (UML) [9]. Indeed, the specification itself can be verified using a model checking technique applied in various domains of computer science. It is possible to check process specification expressed by Petri nets or semi-formal UML activity diagrams. As a result, verified specification is obtained which can provide a base for further steps, e.g. logical synthesis for code implementation.

This paper addresses web content integrity concerns by proposing a new model to verify web content integrity by using Colored Petri Nets (CPNs) simulation; which a backward compatible extension of the concept of Petri nets. CPNs preserve useful properties of Petri nets and at the same time extend initial formalism to allow the distinction between tokens.

There are many approaches established for integrity verification, to assure integrity of the web content. The form-field validation approach protects against harmful data on both the client and server sides. The SSL approach secures the communication channel, and the firewall approach protects against malicious code and other attack strategies [20]. This paper focuses on firewall approaches.

Firewall technology [4] is a mechanism designed to permit or deny network transmissions based on a set of

rules and is used to protect networks from unauthorized access. It has been applied to TCP/IP (Transmission Control Protocol/ Internet Protocol) that protect against outside untrusted connections. There are many firewall architectures that have been published, such as filtering routers. One form of defense for every network connected to the internet is access control lists that reside on routers or firewalls.

Network and application firewalls, [20] provide protection at the host and network levels. These security defenses cannot be used to stop malicious attacks that ask to do illegal transactions. Firewalls are designed to prevent vulnerabilities of signatures and specific ports. Therefore, the firewalls cannot distinguish between the original request-response conversation, and the tampered conversation. They do not track a conversation and do not secure the session information.

In this paper we propose a new model that uses firewall filtering to detect a malicious attack by analyzing abnormal behavior. Also, this model blocks malicious attacks and checks web content integrity synchronously. The aim of this model not only supports maintaining integrity of data, but also ensures that the web content is secured.

The remainder of this paper is organized as follows: Section 2 presents Petri nets an-overview. Section 3 presents related works. Section 4 presents proposed model and simulation. Section 5 presents conclusions.

II. Petri Nets An- Overview

Petri nets (PNs) [13, 26] were invented by Carl Adam Petri in 1962 as part of his Ph.D. It is used to model the functionality and the behavior of system like: the computer system, the computer network and protocols, manufacturing, production, scheduling systems, and controllers. Furthermore, it is used formally for concurrent systems. PNs are a powerful tool for graphical representation and analysis of the software processes, so it used in early stages of the software development process.

PNs are a bipartite graph [13, 26] composed from a place as condition and transition, as event or activity. Places and transitions are connected by directed arcs (edges), and the arcs exist only between a place and a transition or vice versa. It also has tokens which circulate in the system between places. Places are drawn as a circle and represent the passive parts of a system such as buffers, pipes or queues, where transitions are drawn as rectangles and represent the activities in the system such as the failure, repairing, and processing of items. The items are represented by tokens, which are drawn as small black dots and are moved from one place to another along the arcs connecting places and transitions when transitions fire. PNs have many properties such as boundedness, safeness, and being deadlock free.

Petri net is effective for describing and studying information processing systems that are concurrent, asynchronous, distributed, parallel, nondeterministic, and/or stochastic [29, 30]. It therefore enables the analysis of system properties, while a verifying model is enabled rigorously by analyzing the properties and behaviors of Petri nets.

Petri Nets discrete event systems (DES) [11] are working models whose simulation can be observed by Petri Net simulators while mathematically its design correctness of requirements can be proven. So, the mathematical graphical language [16] used to analyze the behavior and properties of the system.

PNs combine the state-event model with the state space are represented by places and events that are represented by transitions. It is used to formalize the behavior of some components, or a system or application, namely those that have a complex behavior.

*Petri net*s [12] are high level and widely used in both theoretical analysis and practical modelling of concurrent systems. It is usually interpreted as a control flow graph of a modeled system. Places correspond to conditions within the system, while transitions correspond to actions. A condition can be fulfilled, i.e. marked with a token in the current system state, or not. An action, i.e. a transition firing, can move tokens between places, thus reflecting a change to the system state.

Petri nets cover a wide class of discrete mathematical models that allow describing control and information flow of the modelled systems [17]. They are an effective facility to model information processing and are used for describing sequential logic circuit behavior. Petri nets are defined by the tuple:

$$PN = (P, T, F, W, M_0) \quad (1)$$

Where $P = \{p_1, p_2, p_3, \ldots. p_n\}$ denotes the set of places, $T = \{t_1, t_2, t_3, \ldots. t_n\}$ denotes the set of transitions, $F \subset (P \times T) \cup (T \times P)$ is the set of directed edges connecting places and transitions, $W:F \rightarrow \{1,2,3,\ldots\}$ is the weight function of the edges and $M_0:P \rightarrow \{1,2,3,\ldots\}$ is the initial marking [17].

Colored Petri nets (CPN) are a backward compatible extension of the concept of Petri nets. CPN preserve useful properties of Petri nets and at the same time extend initial formalism to allow the distinction between tokens [19]. Colored Petri Nets allow tokens to have a data value attached to them. This attached data value is called token color. Although the color can be of an arbitrarily complex type, places in CPNs usually contain tokens of one type. This type is called color set of the place. The formal definition of a colored Petri net is defined by the tuple [19]:

$$CPN = (\sum, P, T, A, N, C, G, E, I) \quad (2)$$

Where $\sum$ non- empty and finite set of types, called

colors, P = {p_1, p_2, p_3,….. p_n} denotes the set of places, T = {t_1, t_2, t_3,….. t_n} denotes the set of transitions, A = P ∩ T = P ∩ A = T ∩ A = ø is a finite set of directed edges connecting places and transitions, N = A to (P × T) ∪(T × P) is the node function, C = P to ∑ is the color function, G is a guard function, that maps each transition t∈T to a guard expression g, the output of the guard expression should evaluate to Boolean value: true or false, E is an arc expression function, it maps each arc a∈A into the expression e, the input and output types of the arc expressions must correspond to the type of the nodes the arc is connected to, I is an initialization function, it maps each place p into an initialization expression i. The initialization expression must evaluate to a multiset of tokens with a color corresponding to the color of the place C(p) [19].

III. Related Works

There are many publications about verification of web content integrity. Most of them focus on the particular stage of the web security protection. Colored Petri nets are used to analyze the software specification.

A. Web Security and Integrity of Data

Double Guard is an application developed by Reddy et al. (2015), is used for checking the intrusions in a multi-tier application. This application is used for back-end and front-end, and it's independent. It is also operated in dynamic and static servers in the web; these servers provide better protection for the application and information [5].

Hidhaya and Angelina (2012), developed a mechanism for the detection of SQL injection by employing a reverse proxy and MD5 algorithm to watch SQL injection in input, which rules of grammar expressions to check SQL injection in URL's. This mechanism has achieved a decrease in the number of attach automatically, decrease the delay of time, and makes it able to protect the application from SQL injection attack [22].

The method of Win, and Htun (2014), is for combined static analysis and runtime validation. Legitimate queries are found in static analysis, can be operated by the application, and can also reform them into patterns of structure query. The query patterns resulting from it are kept in separate respective tables. In order to decrease the runtime validation, overhead of the runtime query in the runtime validation is made into patterns, and a comparison is made between them and the predetermined structure query patterns. The performance of the suggested technique has been evaluated by examining it on weak web applications. The presented method can be performed on both web applications and applications which are linked to a database [28].

Shadi Aljawarneh et al, (2007) have focused on one issue, namely the integrity of web content. It has been shown that given the limitations of SSL, a loss of web content integrity is possible because of the statelessness of HTTP. In an attempt to overcome this problem, we have formulated a systematic web security framework that could provide continued reliable and correct services to external users, even though a web data manipulation problem may have occurred. It was suggested that such a framework will offer an increased level of user confidence, since the framework provides a greater protection against web server subversion [20].

A novel approach for detecting SQL attacks which are based on information theory, was proposed by Hossain Shahriar et al. (2013), is the entropy of all queries, which exists in a program accessed before deploying a program, is computed. During the time of executing the program, this approach depends on the thought that dynamic queries with attack inputs result in a level of entropy that is decreased or increased. Three open source PHP applications which contain SQL weaknesses, proved by report, validated the proposed framework. A prototype tool is implemented by them in Java for facilitating the training and detection phase of the proposed technique. The result of the evaluation indicated that the technique checks all known SQL weaknesses and might be an integral one to verify unknown weaknesses [8].

Gianluca et al. [7] presented redirection graphs to detect malicious webpages; while Kapravelos et al. [1] presented automatically detected evasive behavior using malicious JavaScript.

B. Web Security using Colored Petri Nets

Colored Petri Nets (CPNs) [6, 16] which is an extended version of Petri Nets and are usually used in system modeling, editing, simulating the concurrent systems and analyzing their properties. The major advantages [16] of using CPNs simulation are to gain insight, analysis and specification, and to describe concurrent behavior and interaction behavior of a complex real-time system, which is particularly significant for a safety-critical system. CPNs mainly focus on synchronization, concurrency and asynchronous events [4].

Kumar [21] proposed a network intrusion detection system prototype based on CPNs to describe complicated attacks. The authors [29] studied the knowledge representation and intelligent analysis on aggressive behavior that used the PNs theory to set up related theories and methods, which are suitable for aggressive behavior analysis and detection.

Table 1. Comparison between the Proposed Approach and Previous Solutions

	Intrusion Detection		Integrity Content	
	Detect	Prevent	Detect	Recover
Proposed	Malicious attack	Malicious attack	Tampering	Content
Double Guard [5]	Attack	Attack	--	--
Integrity of web content [20]	--	--	Tampering	Content
SQL injection [22]	SQL injection	--	--	--
Redirection graphs [7]	Malicious webpage	--	--	--
Evasive behavior [1]	Malicious JavaScript	--	--	--

Table 1 compares the proposed approach with multiple previous solutions. The comparison criteria include the ability to detect/prevent attacks, and the ability to detect tampering/recover of web content.

Table 1 shows the proposed solution for detection and prevention of malicious attacks, the detection of tampering and the ability to aid in recovery of web content.

IV. Proposed Model and Simulation

This paper proposes a new model of web content integrity verification using CPNs simulation, focused on achieving two aims:

1. Protecting web security from malicious attack by using a firewall approach.
2. Verifying web content integrity by using detection and recovery techniques.

This model presents the processes of detecting the malicious attack and the recovery of web content.

This section is divided into four main parts: framework, flow chart, web content integrity model, and simulation results.

A. Web Content Integrity Framework

The framework of this paper is divided into two steps. The first step is firewall filtering to protect the web security from malicious attacks. The second step is verifying web content integrity to detect tampering and recovery content.

Figure 1 shows all the steps that make up the framework architecture that is used in this paper for the proposed mechanisms of verification web content integrity. The overall framework consists of the following seven main steps: user request, firewall filter, web server, integrity verifier, detection of tampering, recovery of web content, and showing of the current page.

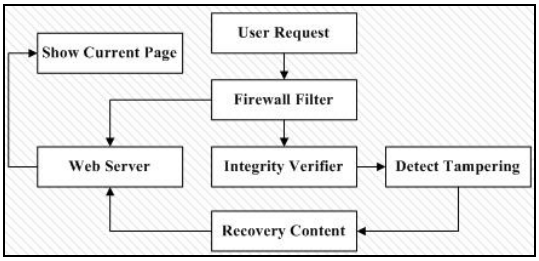

Fig.1. Overall Model Framework Architecture

Our proposed model offers integrity of data, and a higher level of trustworthiness to an organization. We believe that the proposed framework will be capable of verifying web content against tampering. When the firewall finds abnormal behavior, an integrity identifier is enabled to detect and protect web content against tampering, and enable recovery of the original copy of the compromised web content. We are exploring risk analyzer techniques to protect web content.

B. Web Content Integrity Flow Chart

Figure 2 shows the flow chart of model design that consists of two main steps to do the following:

1) Firewall filtering that analyzes the user request packet and filters unauthorized packets by using existing rules in the Access Control List (ACL).
2) Integrity verifier which checks the integrity of web content by comparing the current page with the backup page. If it finds tampering on web content, then it performs a recovery of web content, using a hash technique.

This model presents three paths of testing to insure web content integrity. The first is utilized when no malicious attack is found. The second when a malicious attack is detected which does not affect web content, and the third when malicious attack is found which affects web content.

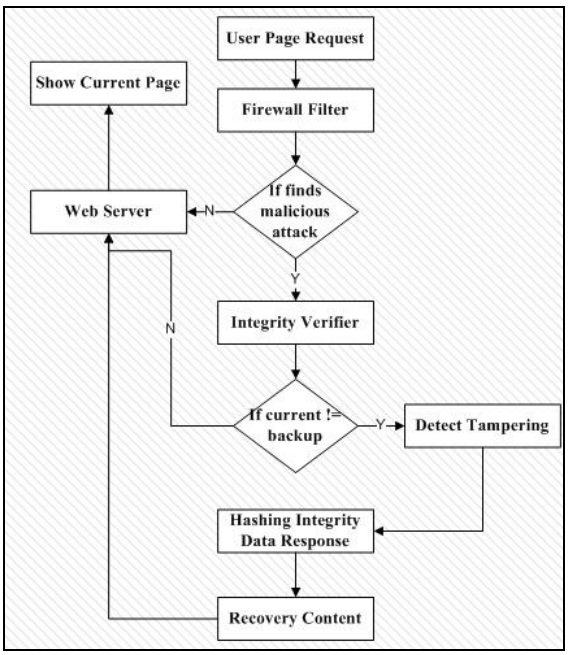

Fig.2. The Flow Chart of Model Design

The proposed model defines algorithm to detect tampering and recovery of web content as the following:

Step 1: Receive packet request for web page into firewall filtering.
Step 2: The firewall analyzing the packet request to check malicious attack.

If

it found out attack

then

It sends the request of page to integrity verifier.

Else

Web server returns the current page of request page.

Step 3: Tthe integrity verifier check the integrity of web content.

If

It detects tampering

Then

Recovers web content and web server returns the current page of request page.

Else

Web server returns the current page of request page.

C. Web Content Integrity Model

The proposed model used firewall filter using ACL rules, and an integrity verifier to protect web content integrity.

Firewall Filtering Model using ACL Rules, fig. 3 shows the model of the firewall. This model filters the incoming transition by using a control unit based on ACL rules to check authorized packets.

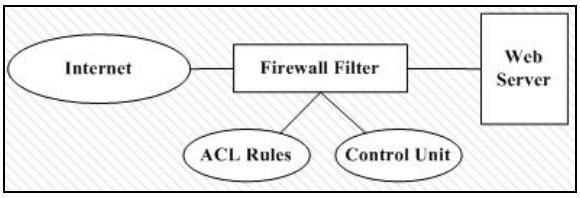

Fig.3. Firewall Model

Integrity Verifier Model, is divided into two sub-models: Detection model using backup method and recovery model using hash method. Figure 4 shows the model of detection mode, and fig. 5 shows the model of recovery mode.

1). Detection Model using Backup Method

This model checks web content integrity by comparing the current page and the backup of page.

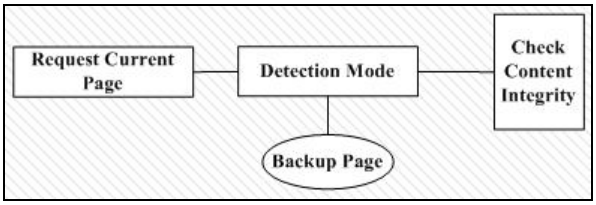

Fig.4. Detection Model

2). Recovery Model using Hash Method

This model restores web content by using a hash page response. Hashing is a technique that aims to ensure the integrity of data by generating unique hash values.

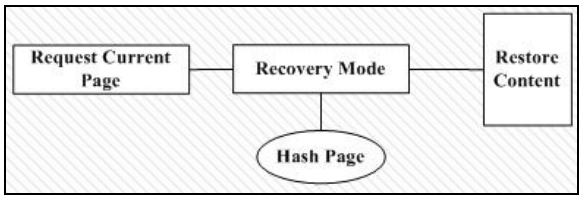

Fig.5. Recovery Model

D. Simulation Results

This paper used CPNs simulation to prove that software specification processes modeled and increased the reliability and quality of the system. The specification processes is formally verified with the proposed model. The model is conducted on two main models of simulation. First, the firewall model shows filtering behavior. Second, the integrity verifier performs two tasks: detects the content tampering and recovers web content.

1). Firewall Filtering Simulation

Figure 6 shows simulation of the behavior of the firewall filtering model when packets are coming. The packet is composed of four parts: MAC of source, MAC of destination, IP of source, and IP of destination. The packets come to the control unit that compares it with the ACL state to check the malicious attack. Then it blocks the unauthorized transition and passes the authorized packet.

The firewall model is composed of two logic rules of test cases:

Rule1, if the firewall detects malicious attack, then block the request.

$$R1: (N1 \wedge N2) \wedge (state = \text{‘true’}) \rightarrow N3$$

Rule2, if the firewall did not detect malicious attack, then pass the request.

$$R2: (N1 \wedge N2) \wedge (state = \text{‘false’}) \rightarrow N4$$

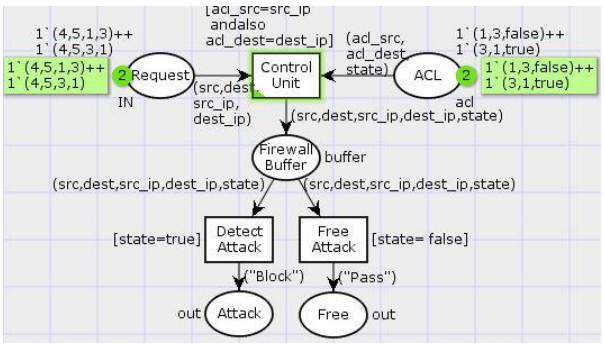

Fig.6. Firewall Filter Simulation

The formalism describes two states on the fig. 6. One in which malicious attack has been detected and the other that is free of attack. The formal definition specified with CPNs simulation that composed of three types, five places, three transitions, seven arcs, four node functions,

six color functions, two main variables, two string constants, three transition expressions, three arcs expressions and two types of tokens.

∑ = the set of types

{Boolean, integer, string}

P = the set of places

{P1 (Request), P2 (ACL), P3 (firewall buffer), P4 (attack), P5 (free)}

T = the set of transitions

{T1 (Control unit), T2 (detect attack), T3 (free attack)}

A = the set of arcs

{A1 (request- control unit), A2 (acl- control unit), A3 (control unit- firewall buffer), A4 (firewall buffer-detect attack), A5 (firewall buffer- free attack), A6 (detect attack- attack), A7 (free attack- free)}

N = the set of node functions

{N1 (A1) ∪ (A3), N2 (A2) ∪ (A3), N3 (A4) ∪ (A6), N4 (A5) ∪ (A7)}

C = the set of color functions

{Color mac = integer; Color ip = integer; Color state = boolean; Color IN = product mac * mac * ip * ip; Color acl = product ip * ip * state; Color buffer = product mac * mac * ip * ip * state; Color out = string;}

Variables =

{V1 (src, dest: mac), V2 (src_ip, dest_ip, acl_scr, acl_dest: ip)}

Constant = {"block", "pass"}
G = it maps each transition (t ∈ T) to a guard expression g

{g1 (acl_src = src_ip andalso acl_dest = dest_ip), g2 (state= 'true'), g3 (state = 'false')}

E = it maps each arc (a ∈ A) into the expression e

{e1 (src, dest, src_ip, dest_ip), e2 (acl_src, acl_dest, state), e3 (src, dest, src_ip, dest_ip, state), e4 (src, dest, src_ip, dest_ip, state), e5 (src, dest, src_ip, dest_ip, state), e6 ("Block"), e7 ("Pass")}

I = the set of an initialization functions

{I1 (P1, P2, P3, P4), I2 (P1, P2, P3, P5)}

2). Integrity Verifier Simulation

Figure 7 shows the simulation of the behavior of integrity verifier model. This simulation presents detection tampering on web content and recovery of web content. The first step checks the web content integrity by comparing page request with backup of page content by detecting the state of content. The next step shows the current page when there is no change on web content and recovers web content if changed using hash page.

The integrity verifier model is composed of two logic rules of test cases:

Rule1, if the integrity verifier did not detect change of web content, then show current page requested.

$$R1: (N1 \wedge N2) \wedge (state = \text{'false'}) \rightarrow N3$$

Rule2, if the integrity verifier detects change of web content, then recover page.

$$R2: (N1 \wedge N2) \wedge (N4 <> N5) \wedge (state = \text{'true'}) \rightarrow N6$$

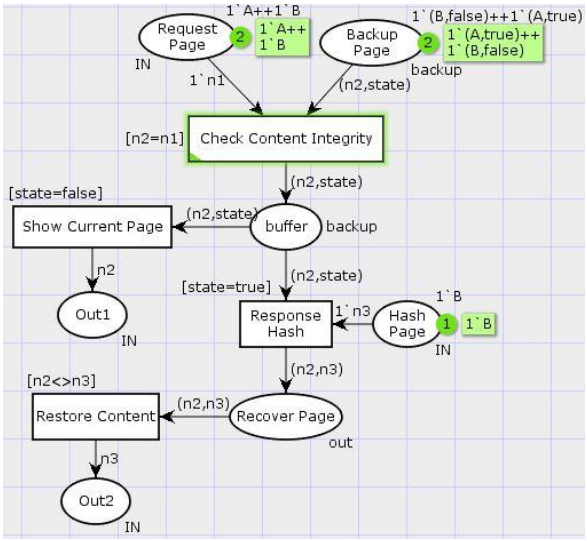

Fig.7. Integrity Verifier Simulation

The formalism describes two states on the fig. 7. One when there is detection of tampering on web content and the other when there is no change found on web content. The formal definition specified with CPN simulation is composed of two types, seven places, four transitions, ten arcs, six node functions, four color functions, two main variables, four transition expressions, ten arcs expressions and two types of tokens.

∑ = the set of types

{Boolean, string}

P = the set of places

{P1 (request page), P2 (backup page), P3 (buffer), P4 (hash page), P5 (recover page), P6 (out1), P7 (out2)}

T = the set of transitions

{T1 (check content integrity), T2 (show current page), T3 (response hash), T4 (Restore content)}

A = the set of arcs

{A1 (request page- check content integrity), A2 (backup page- check content integrity), A3 (check content integrity- buffer), A4 (buffer- show content page), A5 (show content page- out1), A6 (buffer- response hash), A7 (hash page- response hash), A8 (response hash-

recover page), A9 (recover page- restore content), A10 (restore content- out2)}

N = the set of node functions

{N1 (A1) ∪ (A3), N2 (A2) ∪ (A3), N3 (A4) ∪ (A5), N4 (A6) ∪ (A8), N5 (A7) ∪ (A8), N6 (A9) ∪ (A10)}

C = the set of color functions

{Color IN = with A | B; Color state = Boolean; Color out = product IN * IN; Color backup = product IN * state;}

Variables =

{V1 (n1, n2, n3: IN); V2 (state: state);}

G = it maps each transition (t ∈ T) to a guard expression g

{g1 (n2 = n1), g2 (state= 'false'), g3 (state = 'true'), g4 (n2 <> n3)}

E = it maps each arc (a ∈ A) into the expression e

{e1 (1`n1), e2 (n2, state), e3 (n2, state), e4 (n2, state), e5 (n2), e6 (n2, state), e7 (1`n3), e8 (n2, n3), e9 (n2, n3), e10 (n3)}

I = the set of an initialization functions

{I1 (P1, P2, P3, P6), I2 (P1, P2, P3, P4, P5, P7)}

3). Web Content Integrity Simulation

Figure 8 shows the overall proposed model simulation that combines the processes of firewall filtering and integrity verifier in one model. This simulation checks the behavior of a system of three events: malicious attack with detection of a change of web content, malicious attack with no detection of change of web content, and free of attack.

The proposed model is composed of three logic rules of test cases:

Rule1, if the firewall did not detect malicious attack
Then show the current page directly.

R1: (((N1 ^ N2) ^ N4) ^ (state1 = 'false')) → N5

Rule2, if the firewall detects malicious attack and the integrity verifier did not detect change of web content
Then Block the attack and show the current page.

R2: (((N1 ^ N2) ^ N3) ^ (state1 = 'true') → N6 ^
(N7 ← (((N8 ^ N9) ^ (state2 = 'false')) → N10))

Rule3, if the firewall detects malicious attack and the integrity verifier detect change of web content
Then Block the attack and recover page.

R3: (((N1 ^ N2) ^ N3) ^ (state1 = 'true') → N6 ^
(N7 ← (((N8 ^ N9) ^ ((N11 <> N12) ^
(state2 = 'true')) → N13)))

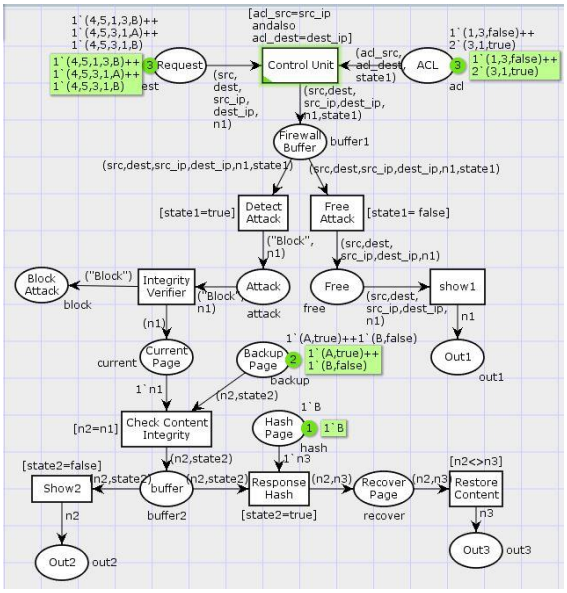

Fig.8. Proposed Model Simulation

The formalism describes three states on the fig. 8. The first is when there is detection of malicious attack with change of web content. The second state is when there is malicious attack without change of web content. The third is when there is no attack detected (free of attack). The formal definition specified with CPNs simulation is composed of three types, fourteen places, nine transitions, twenty two arcs, thirteen node functions, nineteen color functions, four main variables, nine transition expressions, twenty two arc expressions and three types of tokens.

∑ = the set of types

{Boolean, integer, string}

P = the set of places

{P1 (request), P2 (ACL), P3 (firewall buffer), P4 (attack), P5 (free), P6 (out1), P7 (block attack), P8 (current page), P9 (backup page), P10 (buffer), P11 (out2), P12 (hash page), P13 (recover page), P14 (out3)}

T = the set of transitions

{T1 (control unit), T2 (detect attack), T3 (free attack), T4 (show1), T5 (integrity verifier), T6 (check content integrity), T7 (show2), T8 (response hash), T9 (restore content)}

A = the set of arcs

{A1 (request- control unit), A2 (acl- control unit), A3 (control unit- firewall buffer), A4 (firewall buffer- detect attack), A5 (firewall buffer- free attack), A6 (detect attack- attack), A7 (free attack- free), A8 (free- show1), A9 (show1- out1), A10 (attack- integrity identifier), A11 (integrity identifier- block attack), A12 (integrity identifier- current page), A13 (current page- check content integrity), A14 (backup page- check content integrity), A15 (check content integrity- buffer), A16 (buffer- show2), A17 (show2- out2), A18 (buffer- response hash), A19 (hash page- response hash), A20

(response hash- recover page), A21 (recover page- restore content), A22 (restore content- out3)}

N = the set of node functions

{N1 (A1) ∪ (A3), N2 (A2) ∪ (A3), N3 (A4) ∪ (A6), N4 (A5) ∪ (A7), N5 (A8) ∪ (A9), N6 (A10) ∪ (A11), N7 (A10) ∪ (A12), N8 (A13) ∪ (A15), N9 (A14) ∪ (A15), N10 (A16) ∪ (A17), N11 (A18) ∪ (A20), N12 (A19) ∪ (A20), N13 (A21) ∪ (A22)}

C = the set of color functions

{Color mac = integer; Color ip = integer; Color state = Boolean; Color acl = product ip * ip * state; Color S = string; Color con = with A | B; Color request = product mac * mac * ip * ip * con; Color buffer1 = product mac * mac * ip * ip * con * state; Color free = product mac * mac * ip * ip * con; Color out1 = con; Color attack = product S * con; Color block = S; Color current = con; Color backup = product con * state; Color buffer2 = product con * state; Color out2 = con; Color hash = con; Color recover = product con * con; Color out3= con;}

Variables =

{V1 (src, dest: mac); V2 (src_ip, dest_ip, acl_scr, acl_dest: ip); V3 (n1, n2, n3: con); V4 (state1, state2: state);}

Constant = {"block"}
G = it maps each transition (t ∈ T) to a guard expression g

{g1 (acl_src = src_ip andalso acl_dest = dest_ip), g2 (state1= 'false'), g3 (state1 = 'true'), g4 (null), g5 (null), g6 (n1 = n2), g7 (state2 = 'true'), g8 (state2 = 'false'), g9 (n2 <> n3)}

E = it maps each arc (a ∈ A) into the expression e

{e1 (src, dest, src_ip, dest_ip, n1), e2 (acl_src, acl_dest, state1), e3 (src, dest, src_ip, dest_ip, n1, state1), e4 (src, dest, src_ip, dest_ip, n1, state1), e5 (src, dest, src_ip, dest_ip, n1, state1), e6 ("Block", n1), e7 (src, dest, src_ip, dest_ip, n1, state1), e8 (src, dest, src_ip, dest_ip, n1, state1), e9 (n1), e10 ("Block", n1), e11 ("Block"), e12 (n1), e13 (1`n1), e14 (n2, state2), e15 (n2, state2), e16 (n2, state2), e17 (n2), e18 (n2, state2), e19 (1`n3), e20 (n2, n3), e21 (n2, n3), e22 (n3)}

I = the set of an initialization functions

{I1 (P1, P2, P3, P5, P6), I2 (P1, P2, P3, P4, P7), I3 (P1, P2, P3, P4, P8, P9, P10, P11), I4 (P1, P2, P3, P4, P8, P9, P10, P12, P13, P14)}

V. Conclusions

The results shown propose a new model to verify web content integrity by using CPNs simulation. The goal of this model not only supports maintaining integrity of data, but also ensures that the web content is secure. The model achieves two aims: detection tampering and recovery of web content. Furthermore, the model uses firewall filtering to detect malicious attacks. Finally, when the model detects abnormal behavior, it then sends two actions at the same time; blocking the malicious attack and checking the web content integrity.

References

[1] A. Kapravelos, Y. Shoshitaishvili, M. Cova, C. Kruegel, and G. Vigna. 2013. "Revolver: An automated approach to the detection of evasive web-based malware". In Proceedings of the *22nd USENIX Security Symposium*, August 14–16, 2013, Washington, D.C., USA ISBN 978-1-931971-03-4.

[2] Á. Sobrinho, A. Perkusich, L. Silva, and P. Cunha. 2014. "Using Colored Petri Nets for the Requirements Engineering of a Surface Electrogastrography System". *IEEE*, 978-1-4799-4905-2/14/$31.00 ©2014 IEEE.

[3] B. Agarwal. 2013. "Transformation of UML Activity Diagrams into Petri Nets for Verification Purposes". *International Journal Of Engineering And Computer Science* ISSN: 2319-7242 Volume 2 Issue 3 March 2013 Page No. 798-805.

[4] B. Barzegar and H. Motameni. 2011. "Modeling and Simulation Firewall Using Colored Petri Net". *World Applied Sciences Journal* 15 (6): 826-830, 2011 ISSN 1818-4952 © IDOSI Publications, 2011.

[5] D. Reddy, and S. Reddy. 2015. "Detecting Attacks and Protecting From single To Multi Level application". *International Journal of Advanced Technology in Engineering and Science*, Volume No.03, Issue No. 01, pp 478 – 484.

[6] E. Mirzaeian, H. Motameni, S. G. Mojaveri, and A. Farahi. 2010. "An optimized approach to generate object oriented software test case by Colored Petri Net". *2nd International Conference on Software Technology and Engineering (ICSTE)*, 9 78-1-4244-8666-3/10/$26.00 2010 IEEE.

[7] G. Stringhini, C. Kruegel, and G. Vigna. 2013. "Shady paths: leveraging surfing crowds to detect malicious web pages". *CCS* '13, November 4–8, 2013, pages 133–144. Berlin,Germany. Copyright2013ACM978-1-4503-2477-9/13/11. http://dx.doi.org/10.1145/2508859.2516682.

[8] H. Shahriar, S. North, and W. Chen. 2013. "early Detection of SQL Injection Attacks". *International Journal of Network Security & Its Applications*, Vol.5, No.4, pp 53 -65, DOI: 10.5121/ijnsa.2013.5404.

[9] I. Grobelna, R. Wisniewski, M. Grobelny, and M. Wisniewska. 2016. "Design and Verification of Real-Life Processes with Application of Petri Nets". *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2168-2216, 2016 IEEE.

[10] Jose-Inacio Rocha, Luıs Gomes, and Octavio Pascoa Dias. 2011. "Dataflow Model Property Verification Using Petri net Translation Techniques". *IEEE*, 978-1-4577-0434-5/11/$26.00 ©2011 IEEE.

[11] K. Md. Nur. 2011. "Formal Verification of Requirements Engineering Of Road Traffic Control System Using Petri Nets". *Bangladesh Research Publications Journal*, ISSN: 1998-2003, Volume: 5, Issue: 4, Page: 402-411, July - August, 2011.

[12] K. Sacha. 1998. "Safety verification of software using structured Petri nets". Ehrenberger W. (eds) *Computer Safety, Reliability and Security Springer, Berlin, Heidelberg.* vol 1516, https://doi.org/10.1007/3-540-49646-7_26, ISBN 978-3-540-65110-9

[13] L. Peterson. 1977. "Petri Nets". *Computing Surveys*, Vol

9, No. 3, September 1977.
[14] L. Silva, A. Perkusich. 2005. "Composition of software artifacts modelled using Colored Petri nets". *Science of Computer Programming* 56 (2005) 171–189, 0167-6423/$ - see front matter © 2004 Elsevier B.V. All rights reserved. doi:10.1016/j.scico.2004.11.011.
[15] L. Zhua and W. Wang. 2012. "UML Diagrams to Hierarchical Colored Petri Nets: An Automatic Software Performance Tool". *International Workshop on Information and Electronics Engineering* (IWIEE), 1877-7058 © 2011 Published by Elsevier Ltd, doi:10.1016/j.proeng.2012.01.373.
[16] Madhusudanan. J, Anand. P, Hariharan. S, and V. Prasanna Venkatesan. 2014. "Verification of Generic Ubiquitous Middleware for Smart Home Using Coloured Petri Nets". *I.J. Information Technology and Computer Science*, 2014, 10, 63-69, DOI: 10.5815/ijitcs.2014.10.09.
[17] M. Siebert, and J. Flochová. 2013. "PNets - the Verification Tool based on Petri Nets". *World Congress on Engineering* 2013 Vol I, WCE 2013, July 3 - 5, 2013, London, U.K.
[18] Org Desel J. and JuhG. 2001. "What Is a Petri Net?". *Informal Answers for the Informed Reader of the series Lecture Notes in Computer Science*, Volume 2128, pp 1-25.
[19] P. Bon, and S. Collart-Dutilleu. 2013. "From a Solution Model to a B Model for Verification of Safety Properties". *Journal of Universal Computer Science*, vol.19, no. 1(2013), 2-24.
[20] Sh. Aljawarneh, Ch. Laing, and P. Vickers. 2007. "Verification of Web Content Integrity: A new approach to protecting servers against tampering". *http://nrl.northumbria.ac.uk/456*, ISBN: 1-9025-6016-7 © 2007 PGNet.
[21] S. Kumar. 1995. "Classification and Detection of Computer Intrusions". *Phd thesis Department of Computer Sciences*, vol. 19, no. 8, pp. 21-71, 1995.
[22] S. Hidhaya, and A. Geetha. 2012. "Intrusion Protection against SQL Injection Attacks Using a Reverse Proxy". Recent Trends in *Computer Networks and Distributed Systems Security Communications in Computer and Information Science*, Volume 335, pp 252-263, DOI: 10.5121/csit.2012.2314.
[23] T. Mule, A. Mahajan, S. Kamble, and O. Khatavkar. 2014. "Intrusion Protection against SQL Injection and Cross Site Scripting Attacks Using a Reverse Proxy". *(IJCSIT) International Journal of Computer Science and Information Technologies*, Vol. 5 (3), 2014, 2846-2850, ISSN: 0975-9646.
[24] Sherin Hijazi, Amjad Hudaib. 2017. "Using Petri Nets to Verify Design Model: A Survey". *2017 International Conference on Computational Science and Computational Intelligence (CSCI'17)*, IEEE proceeding, DOI: 10.1109/CSCI.2017.174.
[25] Sherin Hijazi, Mahmoud Moshref and Saleh Al-Sharaeh. 2017. "Enhanced AODV Protocol for Detection and Prevention of Blackhole Attack in Mobile Ad Hoc Network". *International Journal of Computers and Technology*, ISSN 2277 – 3061, Volume 16 Number 1, pp 7535 - 7547.
[26] T. Murata. 1989. "Petri Nets: Properties, Analysis and Applications". *IEEE*, Proceedings of the IEEE, VOL. 77, NO. 4, APRIL 1989.
[27] W. Chun-jian, L. Yong-zhi, and X. Fan. 2012. "An Improved Modeling Method Based on Colored Petri Net". *International Conference on Applied Physics and Industrial Engineering*, 1875-3892 © 2011 Published by Elsevier B.V. Selection and/or peer-review under responsibility of ICAPIE Organization Committee. doi:10.1016/j.phpro.2012.02.168.
[28] W. Win, and H. Htun. 2014. "Detection of SQL Injection Attacks by Combining Static Analysis and Runtime Validation". *International Conference on Advances in Engineering and Technology*, Volume 3, Number 20 , pp 95-99 .
[29] X. Li, and D. Li. 2014. "A Network Attack Model based on Colored Petri Net". *Journal of Networks*, vol. 9, no. 7, July 2014.
[30] Y. Harie, and K. Wasaki. 2016. "Formal Verification of Safety Testing for Remote Controlled Consumer Electeonics Using the Petri Net Tool: HiPS". *IEEE 5th Global Conference on Consumer Electronics*, 978- 1 - 5090-2333-2/16.
[31] Y. Xu. 2011. "Modeling and Analysis of Security Protocols Using Colored Petri Nets". *Journal of Computers*, vol. 6, No. 1, January 2011, doi:10.4304/jcp.6.1.19-27.
[32] Z. Xiao-yu, Y. Zhi-jie, and L. Jing-yang. 2016. "Test Generation Approach based on Colored Petri Net of Mode Transition in On-board Subsystem". Proceedings of the *35th Chinese Control Conference* July 27-29, 2016, Chengdu, China.

Authors' Profiles

Sherin Hijazi obtained her bachelor's degree in Management Information Systems from An-Najah University in 2005. She completed her master studies in Computer Information Systems (CIS) from AL- Yarmouk University in 2012. She has 12 years of experience in information systems and programing. She worked in Palestine Securities Exchange as an administrative assistant to the IT department for almost two years. She then moved to the Palestine Technical University Kadoorei (PTUK), Tulkarem- Palestine, where she worked in several positions: Computer Programmer, Head of Programming and finally Lecturer in the department of Applied Computing, 2007- Until now. She is currently a PhD student in Computer Science at the University of Jordan from 2016 and has four scientific publications in various fields of network security, parallel algorithms, information analysis, design, artificial intelligence and representation of knowledge that are her research interests.

Prof. Amjad Ahmad Hudaib obtained his Bc in computer Science from Mutah University in 1991, then he completed his study in a master of computer science from The University of Jordan in 2000, and Ph.D. In Computer Science/ Software Engineering from University of Pisa, Pisa, Italy, he works as Professor in The University of Jordan, form 2016 until now. He interests in Software Engineering, Pattern Matching, Software Architectural Design, Systems Engineering and Tools, Requirements Engineering, Software Testing and Evaluation, Image Processing, Data Mining, Quality Assurance.

How to cite this paper: Sherin Hijazi, Amjad Hudaib,"Verification of Web Content Integrity: Detection and Recovery Security Approach using Colored Petri Nets", International Journal of Computer Network and Information Security(IJCNIS), Vol.10, No.10, pp.1-10, 2018.DOI: 10.5815/ijcnis.2018.10.01

