Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 353 (2020) 5–20

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Performance Modelling of the Impact of Cyber Attacks on a Web-based Sales System

Ohud Almutairi[1] Nigel Thomas[2]

School of Computing
Newcastle University
Newcastle upon Tyne, United Kingdom

Abstract

In this paper we present two performance models of a web-based sales system, one without the presence of an attack and the other with the presence of a denial of service attack. Models are formulated using the PEPA formalism. The PEPA eclipse plug-in is used to support the creation of the PEPA models for the web-based sales system and the automatic calculation of the performance measures identified to evaluate the models. The evaluation of the models illustrates how the performance of the warehouse's sale is negatively affected by denial of service attack through preventing some or all customers' orders from being fulfilled. The resultant delay on selling perishable products would result on products being discarded.

Keywords: Performance Models, PEPA, Cyber-Attacks.

1 Introduction

The growing demand on online sale systems has led to the development of many web-based sale protocols. The online environment suffers from a high security vulnerability, as it is an open medium and this can expose such a system to threats and attacks. A denial of service attack is a cyber attack in which an attacker floods a targeted victim with messages in order to temporarily or permanently prevent legitimate users from accessing resources [1].

Performance has been seen as an important aspect for evaluating such systems. Therefore the impact and cost that cyber attacks contribute to the performance of the system need to be studied. By modelling the performance of such a system, we can better understand how the system behaves in different scenarios, with and without attacks, to provide a sustainable level of performance.

[1] Email: O.M.M.Almutairi2@newcastle.ac.uk

[2] Email: nigel.thomas@newcastle.ac.uk

https://doi.org/10.1016/j.entcs.2020.09.016

1571-0661/© 2020 The Author(s). Published by Elsevier B.V.

The approach used to model a system under investigation in this paper is Performance Evaluation Process Algebra (PEPA). PEPA is a well-known Stochastic Process Algebra (SPA). A system is modelled in the PEPA formalism as a set of components which interact and engage individually or with other components in activities in order to evaluate its performance [4]. Thus, the components represent the active parts in the system and the behaviour of each part is represented by its activities.

In this paper, The description of the system we study is based on Gelenbe and Wang study [3]. We use the PEPA modelling approach to represent how the performance of a warehouse are affected by denial of service attacks on its webserver that is used by the customers to place their order, how the attacks would prevent some or all customers' orders from being fulfilled and how the delay on selling products would result on products being discarded. Also we illustrate how the warehouse sales are disrupted by such an attack on its webserver. We built models of the system with and without the presence of denial of service attack messages. The creation and analysis of PEPA models are supported by the PEPA Eclipse plug-in [7]. This tool has been developed to support Markovian steady-state analysis, stochastic simulation, and Ordinary Differential Equations (ODE) analysis of PEPA models in the Eclipse Platform [7].

The paper is organized as follows. Section 2 presents some existing related studies. In Section 3, the Web-Based Sales system specification without the presence of an attacker's messages and PEPA models are introduced and the proposed model is followed by its evaluation and results. In Section 4, the Web-Based Sales system specification in the presence of an attacker's messages and the proposed PEPA models are introduced and followed by its evaluation and results. Finally, Section 5 concludes the report by providing an overview of the study findings and future work.

2 Related Work

A number of researchers have studied and measured the performance of a system under an attack. For example, Meng et al. studied the performance cost of the security mechanisms of mobile offloading systems under timing attacks in [6]. They pro-posed a hybrid Continuous-time Markov chain (CTMC) and queueing model to ex-plore the performance cost that is introduced by the quantified security attributes. They formulate measures that would optimize the trade-off between security and performance in the studied system. The result of their study presents the best rekeying rate that provides the optimal balance between security and performance for their system.

Furthermore, the performance of an email system under three types of attacks was explored by Wang et al. in [8]. Their study is a model based system, utilizing a queueing network approach. Wang et al. proposed a system model comprised of four queueing models to evaluate the performance of an email security system by subjecting it to attacks. One queueing model represents the email information unit and each of the remaining models represent one attack. The effects on the system when under the following attacks was noted: mail bombs such as Denial of Service attacks, Password Cracked and a malicious mail attack that contains, for example, a Trojan Horse. The study evaluates the trade-off between performance and security system according to three-proposed metrics: system availability and information leakage probability as a security metrics and average queue length as a performance metric. The numerical findings from the study show how effective and efficient their approach is when analysing the security aspect of email systems under attack.

Gelenbe and Wang studied the performance of warehouse in the presence of a denial of service attack on its webserver [3]. They focused on the economic performance of the warehouse that sells perishable products. They predicted the income loss which results from such an attack. A queuing theory based technique was applied in their study and their approach is suggested to be used as optimization problem to such a system. Zhu and Martinez studied a resilient control problem for linear systems which is subject to replay attacks [9]. They studied the influence of replay attacks on a system stability and performance. Model predictive control method was employed in their study.

Finally, although many studies have been conducted, more research is needed regarding modelling and investigating the performance cost introduced by the attacks in a system. It can be used to better understand how the system behaves and adapts in different scenarios (with and without a threat) to remain secure and to provide a sustainable level of performance.

3 Web-Based Sales System

3.1 System Specification

The description of the system with no attack is as follows [3] (the words in bold are the actions name that we used in our proposed PEPA model):

- A warehouse has an online webserver to sell perishable products.
- The perishable products have a limited shelf life.
- Products arrive from supplier to warehouse at rate s1 (**objectArrive**).
- The warehouse's webserver receives the orders from customers at rate r1 order per time unite (**order**).
- Webserver forwards the successful order messages to the warehouse at rate r2 (**forwardOrder**).
- The expiry date of perishable products will be checked periodically in warehouse (**checkObject**).
- The two ways to remove the products from the warehouse are:
  - Products are removed due to its perishability (**removeExpiredObject**).
  - Products are removed after they are successfully sold (**removeSoldObjects**).
- If the warehouse does not have the product that the customer ordered, then the warehouse will order it from a supplier and pay for it (**requestObject**, **pay-**

ForObject). Then the product will be delivered to warehouse (**objectArrive**). We assume that the warehouse already has some products.

3.2 PEPA Model of the System without Attacks

In our PEPA model, there are five types of components: Customer, Webserver, Warehouse, Timer and Supplier. The model comprises of 5 parts, one for each com-ponent. Customer, Webserver, Warehouse and Timer move sequentially from their different behaviours based on the activities specified in the model. The model is formulated as follows:

Customer component

$$Customer_0 \overset{def}{=} (order, c_1).Customer_1$$
$$Customer_1 \overset{def}{=} (orderSucessfullyPlaced, r_1).Customer_2$$
$$Customer_2 \overset{def}{=} (complete, c).Customer_0$$

This part of the model specifies customer's different behaviours, moving from Customer0 to Customer2. First state is Customer0. It is when customer visits a webserver and performs action order at rate c1 leading to Customer1. Then, in state Customer1 the only action happens is orderSucessfullyPlaced at rate r1 leading to Customer2 which is the state when customer performs action complete at rate c leading back to Customer0 which it means that the interaction between customer and webserver has finished. After Customer2, the behaviour returns to Customer0 so that the model becomes cyclic and steady state measures can be obtained.

Webserver component

$$Webserver_0 \overset{def}{=} (order, c_1).Webserver_1$$
$$Webserver_1 \overset{def}{=} (processingOrder, r_3).Webserver_2$$
$$Webserver_2 \overset{def}{=} (orderSucessfullyPlaced, r_1).Webserver_3$$
$$Webserver_3 \overset{def}{=} (forwardOrder, r_2).Webserver_0$$

Above component represents webserver's different behaviours, moving from Webserver0 to Webserver3. First state is Webserver0. When webserver in state Web-server0 , webserver performs action order to receive an order from customer at rate c1 leading to Webserver1. Then, in state Webserver1, the only action happens is pro-cessingOrder at rate r3 leading to Webserver2. In state Webserver2, the only oreder-SucessfullyPlace can occur at rate r1 leading to Webserver3 which is the state when the webserver forwards the successful order to the warehouse at rate r2.

Warehouse component

$$\begin{aligned}
Warehouse_0 &\stackrel{def}{=} (forwardOrder, r_2).Warehouse_1 \\
&+ (checkObject, w_8).Warehouse_8 \\
&+ (objectArrive, s_1).Warehouse_0 \\
Warehouse_1 &\stackrel{def}{=} (objectExist, w_1).Warehouse_5 \\
&+ (objectNotExist, w_2).Warehouse_2 \\
Warehouse_2 &\stackrel{def}{=} (requestObject, w_3).Warehouse_3 \\
Warehouse_3 &\stackrel{def}{=} (payForObject, w_4).Warehouse_4 \\
Warehouse_4 &\stackrel{def}{=} (objectArrive, s_1).Warehouse_5 \\
Warehouse_5 &\stackrel{def}{=} (addObjectToOrderList, w_5).Warehouse_6 \\
&+ (addObjectToOrderList, w_6).Warehouse_1 \\
Warehouse_6 &\stackrel{def}{=} (removeSoldObjects, w_7).Warehouse_7 \\
Warehouse_7 &\stackrel{def}{=} (sendOrderToCustmer, w_{12}).Warehouse_0 \\
Warehouse_8 &\stackrel{def}{=} (expired, w_9).Warehouse_9 \\
&+ (notExpired, w_{10}).Warehouse_0 \\
Warehouse_9 &\stackrel{def}{=} (removeExpiredObject, w_{11}).Warehouse_0
\end{aligned}$$

The above part of the model is for warehouse component. In state Warehouse0, one of three actions could happen either forwardOrder at rate r2 leading to Warehouse1 if warehouse receives an order from webserver, checkObject at rate w8 leading to Warehouse8 when warehouse checks periodically the expiry date of the products, or objectArrive at rate s1 when some products arrives to warehouse from supplier leading back to Warehouse0. In state Warehouse1, either actions could be performed objectExist at rate w1 leading to Warehouse5 or objectNotExist at rate w2 leading to Warehouse2 in order to request the product from a supplier. In state Warehouse2, the only action happens is requestObject at rate w3 leading to Warehouse3. Then in state Warehouse3, there is only one action could happen which is payForObject at rate w4 leading to Warehouse4 which is the state when warehouse receive the object from supplier after performing action objectArrive at rate s1 leading to Warehouse5. In state Warehouse5, one of two action could be performed either addObjectToOrderList at w5 leading to Warehouse6 in order to remove these product from the warehouse or addObjectToOrderList at rate w6 leading to Warehouse1 when there is more than one product in the order. In Warehouse6, the only action happens is removeSoldObjects at rate w7 leading to Warehouse7 which is the state when the warehouse sends the order to the customer by performing action sendOrderToCustomer at rate w12 leading back to Warehouse0. In Warehouse8, one of two action can be performed either expired at rate w9 leading to Warehouse9 in order to remove the expired product from the warehouse or notExpired at rate w10 leading back to Warehouse0. Finally in state Warehouse9, the only action happens is removeExpiredObject at rate w11 leading back to Warehouse0.

Timer component

$$\begin{aligned}
Timer_0 &\overset{def}{=} (tick, t).Timer_1\\
&+ (notExpired, w_{10}).Timer_0\\
&+ (sendOrderToCustmer, w_{12}).Timer_0\\
Timer_1 &\overset{def}{=} (tick, t).Timer_2\\
&+ (notExpired, w_{10}).Timer_1\\
&+ (sendOrderToCustmer, w_{12}).Timer_0\\
Timer_2 &\overset{def}{=} (expired, w_9).Timer_0\\
&+ (sendOrderToCustmer, w_{12}).Timer_0
\end{aligned}$$

This part of the model is for timer component which is the part that count down the time life of the product in the warehouse. It has 3 behaviours starting form Timer0. In Timer0, one of three actions can happen either tick at rate t to count down the time life of the product in the warehouse leading to Timer1, action notExpired at rate w10 leading back to Timer0 or action sendOrderToCustomer at rate w12 when the product is sent to customer before reaching expiry date leading back to Timer0. Timer1 state has the same action as in Timer0. In state Timer2, one of two actions can be performed either expired when the product expired at rate w9 leading back to Timer0 or action sendOrderToCustomer at rate w12 when the product is sent to customer before reaching expiry date leading back to Timer0.

Supplier component

$$Supplier \overset{def}{=} (requestObject, w_3).(payForObject, w_4).(objectArrive, s_1).Supplier$$

The last part of the model is for supplier component. There is just one state which is Supplier. The supplier's main action is objectArrive .The rate of this action is controlled by supplier.

System equation

The system equation and complete specification are given by

$$System \overset{def}{=} (Customer_0[N] \underset{\mathcal{K}}{\bowtie} Webserver_0[N] \underset{\mathcal{L}}{\bowtie} Warehouse_0 \underset{\mathcal{S}}{\bowtie} (Supplier \| Timer_0)$$

Where K={order, orderSucessfullyPlaced}, L={forwardOrder}, S={requestObject, payForObject, objectArrive, notExpired, expired, sendOrderToCustmer}, any action in list K, L and S is a shared action between the components specified in system equation. N is the number of Customer and Webserver instances in the system. The Five components are initially in the states

Customer0, Webserver0, Warehouse0, Supplier and Timer0.

3.3 Performance evaluation and results

The current investigation seeks to calculate throughput of some main actions of the Webserver and Warehouse which are orderSucessfullyPlaced and forwardOrder actions of the Webserver, and removeSoldObjects and removeExpiredObject actions of the Warehouse using ODE. Throughput is the amount of work that can be accomplished in a specific time [5]. Moreover, we want to investigate on the population level of some states of the components using ODE. The population is the average number of the component's copies in a state waiting for an action in the model. We keep the number of customer to 100. All the action rates for the customer is 1 except complete action is 0.01 to let customer wait and use the product before starting again. All the action rates of the webserver is 100 and is divided by the number of the customers using the webserver except forwardOrder action which we assume to be faster and have a rate of 200 which also depends on the number of customers in webserver. All the action rates of the warehouse is 100 except sendOrderToCustmer action, which we assume to be slower and it has a rate value of 24, and the rates of checkObject and objectArrive actions, which we assume they have rate values equal to forwardOrder action's rate to avoid action race as they are in a same state Warehouse0 in the model. The action rate for timer component that count down the time life of the products in the warehouse is 24.

As can be seen from following graphs (Fig. 1 and 2), the throughput values of the actions orderSucessfullyPlaced, forwardOrder and removeSoldObjects are larger than the throughput value of removeExpiredObject. This indicates that larger products are sold and less products are discarded when there is no attack in the system.

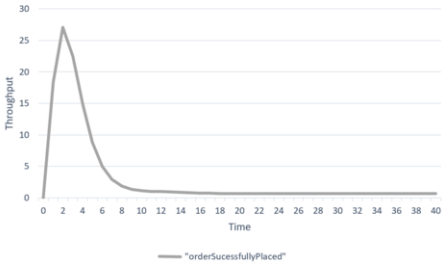

Fig. 1. The Throughput Analysis of orderSucessfullyPlaced.

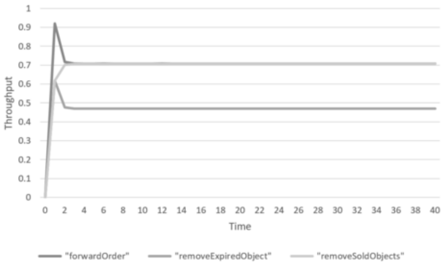

Fig. 2. The Throughput Analysis of forwardOrder, removeExpiredObject and removeSoldObjects.

The following two graphs are the population level analysis for the following states: Customer1 (the customer's state when waiting to receive a successful order confirmation message from webserver), Webserver3 (the webserver's state when waiting to forward the successful order to the warehouse), Warehouse6 (the warehouse's state when removes sold products from warehouse) and Warehouse9 (the warehouse's state when removes expired products from warehouse due to their perishability). As you can see from Fig. 3 and 4, all the customers have their orders forwarded to the warehouse. Fig. 4 illustrates that, in the warehouse, the amount of products that were sold is larger than the amount of products that were discarded.

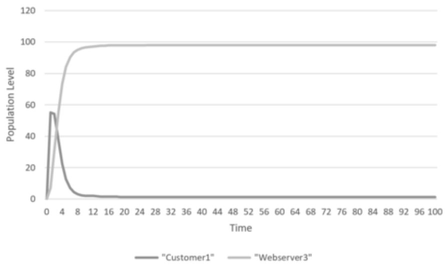

Fig. 3. The population level analysis of Customer1 and Webserver3.

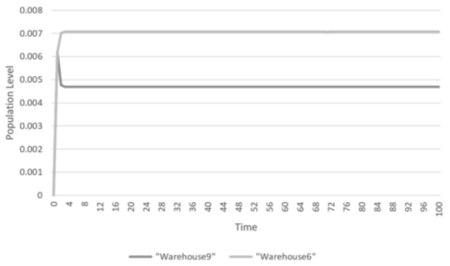

Fig. 4. The population level analysis of Warehouse6 and Warehouse9.

4 Web-Based Sales system in the presence of the attacks

4.1 System Specification

Now we are going to introduce attack to the system. The kind of attack that we want to consider is denial of service attack, whereby an attacker floods the system with bogus requests. The attacker behaves in a similar way to a negative customer in G-networks [2].The effect of this increase in traffic is to delay the processing of legitimate orders. If this delay is sufficiently long then orders will fail due to timeouts and hence revenue will be lost and goods may perish. A realworld denial of service attack may cause systems to overflow resulting in message loss, although we do not model this aspect of the attack. Our aim in modelling this form of attack is to investigate the relationship between the rate of attack and the impact on order throughput and products perishing in the warehouse.

The system has the same steps as in Section 3.1 and the following are steps of introducing the attack to the system [3](the words in bold are the actions name used in our proposed PEPA model):

- The webserver receives an attack message at some rate a1 per time unit based on their vulnerability to attack message with a probability q (**attackM**).
- Each attack message destroy one processing order (**destroyedOrder**).
- The customer that has a destroyed order could repeat the same order with a probability p (**repeatOrder**).

We assume that the number of attack message is equal to the number of website copy created for each customer. If the website is vulnerable to attack message the current processing order will be destroyed.

4.2 PEPA Model of the System in the Presence of the Attacks

In our PEPA model, there are six types of components: Customer, Attacker messages, Webserver, Warehouse, Timer and Supplier. The model comprises of 6 parts, one for each component. Customer, Attacker messages, Webserver, Warehouse and Timer move sequentially from their different behaviours based on the activities specified in the model. The model is formulated as follows:

Customer component

$$
\begin{aligned}
Customer_0 &\overset{def}{=} (order, c_1).Customer_1\\
Customer_1 &\overset{def}{=} (orderSuccessfullyPlaced, r_1).Customer_3\\
&\quad + (destroyedOrder, a_2 * p).Customer_2\\
&\quad + (destroyedOrder, a_2 * (1-p)).Customer_3\\
Customer_2 &\overset{def}{=} (repeatOrder, c_2).Customer_1\\
Customer_3 &\overset{def}{=} (complete, c).Customer_0
\end{aligned}
$$

This part of the model specifies customer's different behaviours. First state is same as customer first's state in Section 3.2. Then, in stateCustomer1, one of three actions can happen either orderSucessfullyPlaced at rate r1 leading to Customer3, destroyOrder at rate a2*p leading to Customer2 or destroyOrder at rate a2*(1-p) lead-ing to Customer3 which is the state when customer performs action complete at rate c leading back to Customer0 which it means that the interaction between customer and webserver has finished. In Customer2, the only action happens is repeatOrder at rate c2 leading back to Customer1.

AttackerM component

$$
\begin{aligned}
AttackerM_0 &\overset{def}{=} (attackM, a_1).AttackerM_1\\
AttackerM_1 &\overset{def}{=} (destroyedOrder, a_2).AttackerM_0
\end{aligned}
$$

Above component represents the two attacker message's behaviours. First state is AttackerM0. It is when the attacker sends an attack message by performing attackM action at rate a1 leading to AttackerM1. The second state is AttackerM1. It is when the attacker message destroys the current processing order based on webserver's vulnerability to attack message by performing destroyedOrder at rate a2 leading back to AttackerM0.

Webserver component

$$
\begin{aligned}
Webserver_0 &\stackrel{def}{=} (order, c_1).Webserver_1\\
&\quad + (repeatOrder, c_2).Webserver_1\\
Webserver_1 &\stackrel{def}{=} (processingOrder, r_3 * q).Webserver_4\\
&\quad + (processingOrder, r_3 * (1-q)).Webserver_2\\
Webserver_2 &\stackrel{def}{=} (orderSucessfullyPlaced, r_1).Webserver_3\\
Webserver_3 &\stackrel{def}{=} (forwardOrder, r_2).Webserver_0\\
Webserver_4 &\stackrel{def}{=} (attackM, a_1).Webserver_5\\
Webserver_5 &\stackrel{def}{=} (destroyedOrder, a_2).Webserver_0
\end{aligned}
$$

Above component of the model is for webserver's different behaviours. The local states of the webserver component increase to 5 states compare to the webserver component when there is no attack (Section 3.2). First state isWebserver0. When webserver in state Webserver0 , webserver performs one of two actions either order to receive an order from customer at rate c1 leading to Webserver1 or repeatOrder at rate c2 leading also to Webserver1. Then, in state Webserver1, one of two actions can happen either processingOrder at rate r3*q leading to Webserver4 or processingOrder at rate r3*(1-q) leading to Webserver2. States Webserver2and Webserver3 are same as webserver's states in Section 3.2. In state Webserver4, the only attackM can occur at rate a1 leading to Webserver5 which is the state when an current processing order is destroyed in webserver by performing destroyedOrder at rate a2.

Other components Warehouse, Timer and Supplier components have same behaviours as in Section 3.2.

System equation

The system equation and complete specification are given by

$$System \stackrel{def}{=} ((Customer_0[N] \underset{\mathcal{H}}{\bowtie} AttackerM_0[N]) \underset{\mathcal{K}}{\bowtie} Webserver_0[N] \underset{\mathcal{L}}{\bowtie} Warehouse_0 \underset{\mathcal{S}}{\bowtie} (Supplier \| Timer_0))$$

Where H= {destroyedOrder}, K= {order, orderSucessfullyPlaced, destroyedOrder, repeatOrder, attackM}, L={forwardOrder}, S={requestObject, payForObject, objectArrive, notExpired, expired, sendOrderToCustmer} any action in list H, K, L and S is a shared action between the components specified in system equation. N is the number of Customer, AttackerM and Webserver instances in the system. The six components are initially in the states Customer0, AttackerM0, Webserver0, Warehouse0, Supplier and Timer0.

4.3 Performance evaluation and results for the extended model

We are seeking to illustrate how the performance of the warehouse's sale would be affected by denial of service attack through preventing some or all customers' orders from being fulfilled. Then the delay on selling perishable products would result on products being discarded. So the warehouse would lose customers and waste products which would also affect the warehouse economically. All the same actions as in Section 3.2 have same rate values as specified in Section 3.3. However, we assume all AttackerM's actions' rate is 4 which is faster than customer's rates. Also, we assume that a customer that has a destroyed order can repeat the same order with a probability of 0.5 (50%).

Our investigation seeks to calculate throughput of some main actions of Customer, AttackerM, Webserver and Warehouse using ODEs. These actions are repeatOrder action of customer, destroyedOrder action of attackerM, orderSucessfullyPlaced and forwardOrder actions of Webserver, and removeSoldObjects and removeExpiredObject actions of Warehouse. Moreover, we want to investigate on the population level of some states of the components using ODE. We assume all perishable products are fresh in the beginning of the simulation then the Timer component counts down its life time.

Throughput analysis

The following graphs show the throughput values of repeatOrder, destroyedOrder, orderSucessfullyPlaced, forwardOrder, removeSoldObjects and removeExpiredObject actions. The throughput values of the actions were calculated based on changing the probability of the webserver to be vulnerable to the attacker's message to 90%, 50%, and then 10%. Figures 5 to 12 show how increasing the probability of the webserver to be vulnerable to the attacker's message was result on more orders were destroyed, sale delay and then more products were discarded due to their perishability. The throughput values of destroyedOrder and removeExpiredObject are larger than orderSucessfullyPlaced and removeSoldObjects when the probabilities of the webserver to be vulnerable to the attacker's message are 90% and 50%. They also show that the larger the probability of the webserver to be vulnerable to the attacker's message, the faster the system lose its control to deal with the orders and attacker's messages which then causing delay on the sale due to preventing some or all customers' orders from being fulfilled and then increasing the amount of products that have been discarded.

In Fig. 5, 6 and 7, the probability of the webserver to be vulnerable to the attacker's message is 90% (q=0.9). In Fig. 5, the throughput value of destroyedOrder starts larger than the throughput value of orderSucessfullyPlaced for period of time. This means many orders are prevented from being fulfilled due to the attacks. In Fig. 6 and 7, the throughput value of removeExpiredObject starts less than the throughput values of forwardOrder and removeSoldObjects and then start to increase during a short period of time. This indicates that the delay on sale the products cause larger amount of products being discarded due to their limited shelf life.

In Fig. 8, 9 and 10, the probability of the webserver to be vulnerable to the attacker's message is 50% (q=0.5). Decreasing the probability of the webserver to be vulnerable to the attacker's message causes the throughput value of destroyedOrder

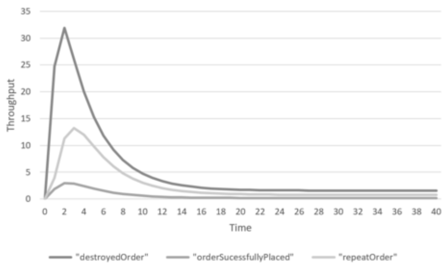

Fig. 5. The Throughput Analysis of destroyedOrder, orderSucessfullyPlaced and repeatOrder when q=0.9.

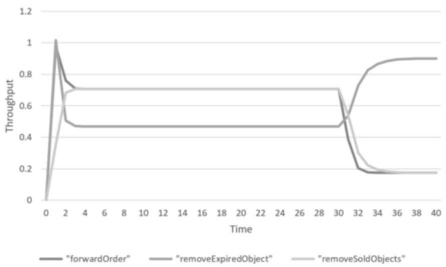

Fig. 6. The Throughput Analysis of forwardOrder, removeExpiredObject and removeSoldObjects when q=0.9.

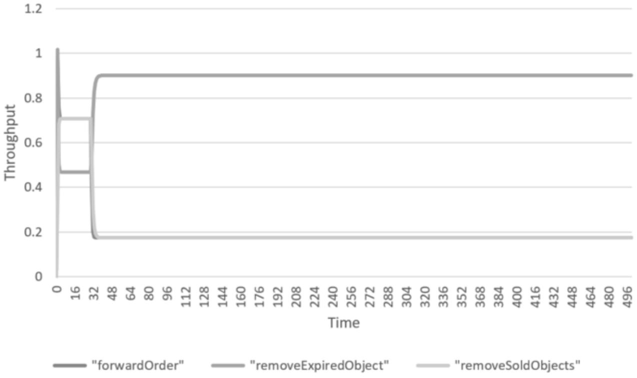

Fig. 7. The Throughput Analysis of forwardOrder, removeExpiredObject and removeSoldObjects in longer time when q=0.9.

to decrease and the throughput value of orderSucessfullyPlaced to increase as in Fig. 8 compare to Fig. 5 when the probability is larger. Moreover, when the probability of the webserver to be vulnerable is larger as in Fig. 6 and 7, the system loses it control to deal with the customer order and attacks faster compare to Fig. 10 when the probability is decreased.

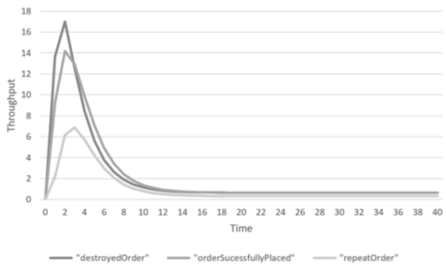

Fig. 8. The throughput analysis of destroyedOrder, orderSucessfullyPlaced and repeatOrder when q=0.5.

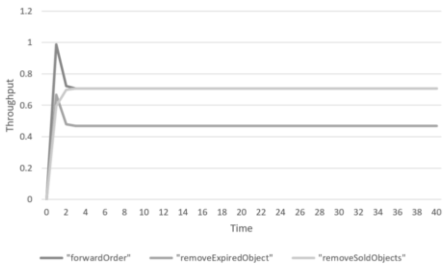

Fig. 9. The throughput analysis of forwardOrder, removeExpiredObject and removeSoldObjects when q=0.5.

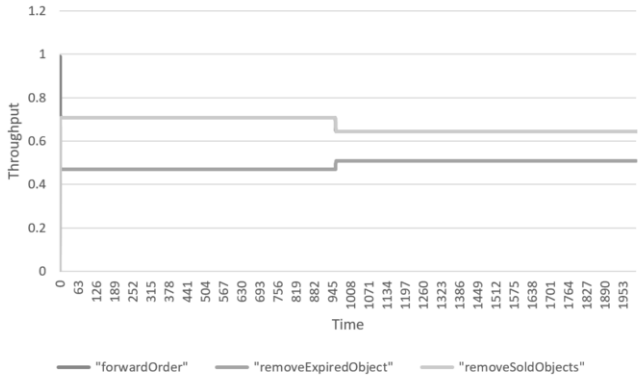

Fig. 10. The throughput analysis of forwardOrder, removeExpiredObject and removeSoldObjects in longer time when q=0.5.

In Fig. 11 and 12, the probability of the webserver to be vulnerable to the attacker's message is 10% (q=0.1). When the probability of the webserver to be vulnerable is low, the larger number of customers' order is fulfilled. The throughput values of the actions orderSucessfullyPlaced, forwardOrder and removeSoldObjects are larger than the throughput value of destroyedOrder and removeExpiredObject.

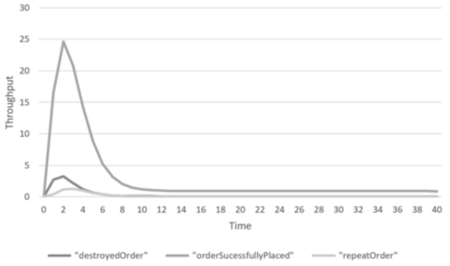

Fig. 11. The throughput analysis of destroyedOrder, orderSucessfullyPlaced and repeatOrder when q=0.1.

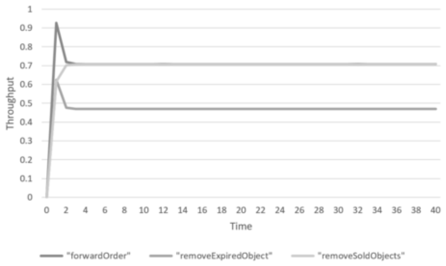

Fig. 12. The throughput analysis of forwardOrder, removeExpiredObject and removeSoldObjects when q=0.1.

Moreover, Fig.13 and 14 shows the different throughput values of destroyedOrder, orderSucessfullyPlaced, repeatOrder, forwardOrder, removeExpiredObject and removeSoldObjects actions in relation to different probability values (q) of the webserver to be vulnerable to the attacker's message. It illustrates how increasing q would have a significant impact in the throughput values of the actions.

Population level analysis

The following graphs are the population level analysis for the following states: Customer2 (the customer's state when waiting to repeat their destroyed order), Webserver2 (the webserver's state when waiting to forward the successful order to the warehouse), Webserver5 (the webserver's state when the order is destroyed), Warehouse6 (the warehouse's state when removing sold products from warehouse) and warehouse9 (the warehouse's state when removing expired products from warehouse due to their perishability). Figures 15 to 20 illustrate how increasing the

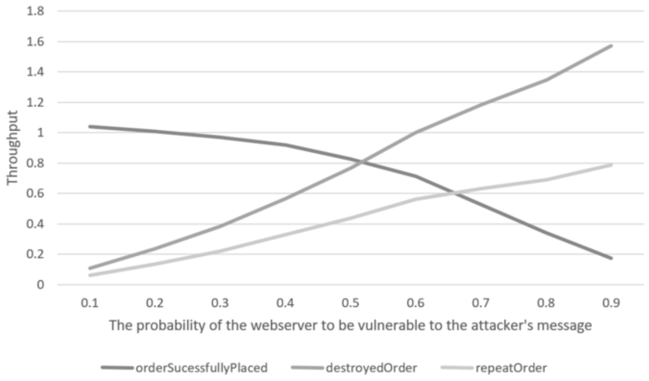

Fig. 13. The throughput analysis of destroyedOrder, orderSucessfullyPlaced and repeatOrder in relation to q different values.

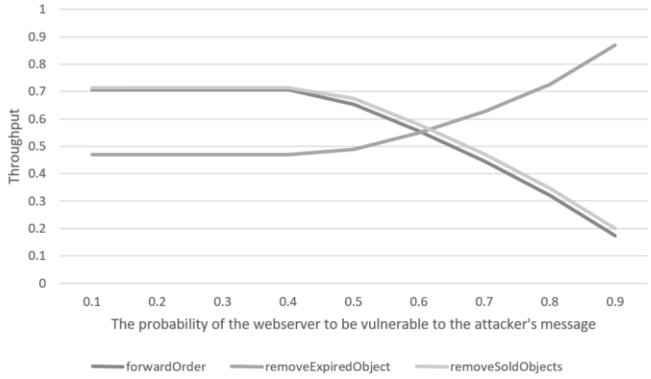

Fig. 14. The throughput analysis of forwardOrder, removeExpiredObject and removeSoldObjects in relation to q different values.

probability of the webserver to be vulnerable to the attacker's message caused the average number of Warehouse6 and Webserver2 copies to decrease and Warehouse9, Customer2 and Webserver5 to increase. In Fig. 15 and 16, the probability of the webserver to be vulnerable to the attacker's message is 90% (q=0.9).In Fig. 17 and 18, the probability is 50% (q=0.5).In Fig. 19 and 20, the probability is 10% (q=0.1).

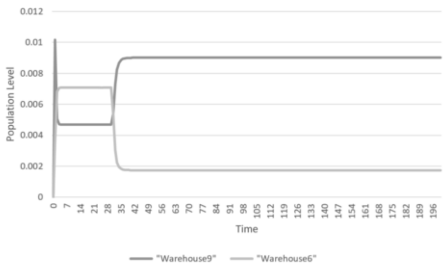

Fig. 15. The population level analysis of Warehouse6 and Warehouse9 when q=0.9.

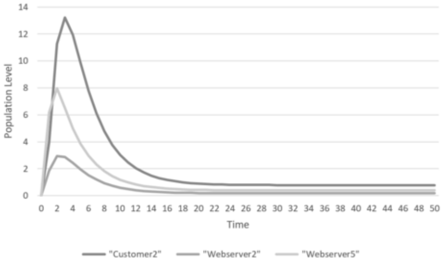

Fig. 16. The population level analysis of Customer2, Webserver2 and Webserver5 when q=0.9.

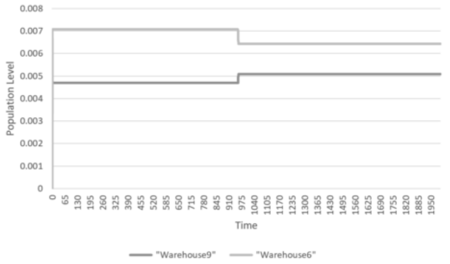

Fig. 17. The population level analysis of Warehouse6 and Warehouse9 when q=0.5.

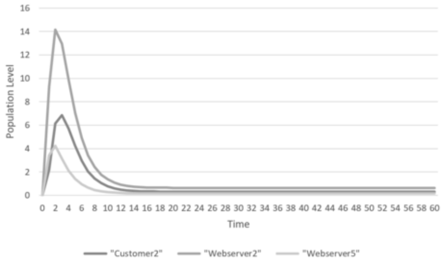

Fig. 18. The population level analysis of Customer2, Webserver2 and Webserver5 when q=0.5.

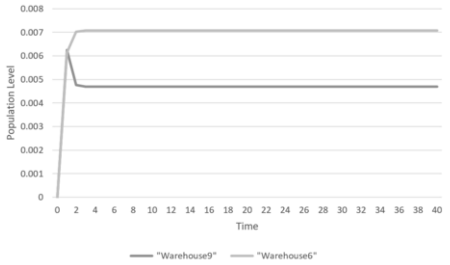

Fig. 19. The population level analysis of Warehouse6 and Warehouse9 when q=0.1.

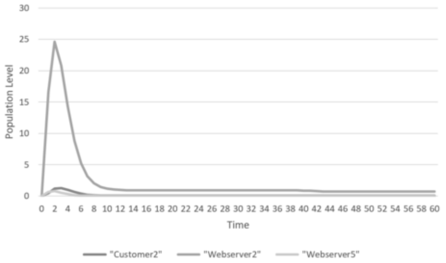

Fig. 20. The population level analysis of Customer2, Webserver2 and Webserver5 when q=0.1.

Moreover, in order to make the system less vulnerable to the attacks, some work is needed to be done to protect the system and that might slow down the system. Therefore, varying q rate could have some other implications in the system, for example, slowing down the processing of the orders.

5 Conclusion and further work

In this paper, we presented the performance models of web-based sale system in two scenarios, with and without the presence of denial of service attacks. It is clear from the related work that there are few existing models in this domain and there-fore this work is a potentially valuable addition. Models are formulated using the PEPA formalism. The proposed performance models illustrated the high level interaction between the components. This study used a PEPA eclipse plug-in to support the creation and evaluation of the proposed PEPA models. The parameters used are somewhat arbitrary, nevertheless the evaluation of the throughput and population level of the proposed models shows how the attacks would prevent some or all positive customers' orders from being fulfilled and how the delay on selling products would result on products being discarded.

Clearly the rates of actions used in the model will have an impact on the effect of any attack. It is clearly therefore desirable to obtain more realistic parameter values from a real system and thereby validate the model. However, even without this, the model clearly demonstrates the impact of a denial of service attack on this system. This means that the model could be extended to explore the cost and benefit of potential defensive mechanisms and further understand the performance security trade-off in this context.

References

[1] Tom Anderson, Timothy Roscoe, and David Wetherall. "Preventing Internet denial-of-service with capabilities". In: *ACM SIGCOMM Computer Communication Review* 34.1 (2004), pp. 39–44.

[2] Erol Gelenbe. "Product-form queueing networks with negative and positive customers". In: *Journal of applied probability* 28.3 (1991), pp. 656–663.

[3] Erol Gelenbe and Yi Wang. "Modelling the impact of cyber-attacks on web based sales". In: *Submitted for Publication* (2019).

[4] Jane Hillston. *A compositional approach to performance modelling*. Vol. 12. Cambridge University Press, 2005.

[5] IEEE. *IEEE Standard Glossary of Software Engineering Terminology*. IEEE, 1990. DOI: 10.1109/IEEESTD.1990.101064.

[6] Tianhui Meng, Katinka Wolter, and Qiushi Wang. "Security and performance tradeoff analysis of mobile offloading systems under timing attacks". In: *European Workshop on Performance Engineering*. Springer. 2015, pp. 32–46.

[7] Mirco Tribastone, Adam Duguid, and Stephen Gilmore. "The PEPA eclipse plug-in". In: *Performance Evaluation Review* 36.4 (2009), p. 28.

[8] Yang Wang, Chuang Lin, and Quan-Lin Li. "Performance analysis of email systems under three types of attacks". In: *Performance Evaluation* 67.6 (2010), pp. 485–499.

[9] Minghui Zhu and Sonia Martinez. "On the performance analysis of resilient networked control systems under replay attacks". In: *IEEE Transactions on Automatic Control* 59.3 (2013), pp. 804–808.