Internet Payment Security and a Case Study

Hortensia Mena
Internet Payment Security and a Case Study

Hortensia Mena

Master’s Thesis in Computer Science (30 ECTS credits)
First Degree Programme in Mathematics – Natural Science
Stockholm University year 2015
Supervisor at NADA was Sonja Buchegger
Examiner was Johan Håstad

Department of Numerical Analysis and Computer Science
Royal Institute of Technology, School of Computer Science and Communication
SE-100 44 Stockholm, Sweden
Abstract

Internet payments are increasing. People are not only buying physical goods on the internet but also virtual goods such as tickets, lottery tickets and virtual money. It is becoming an important part of the economy.

In order for ecommerce to increase even more it is essential that consumers trust the security of the system. The confidentiality of the consumer card details has to be guaranteed. The merchant needs to know that he sends the goods to the holder of the card and not an imposter. The card issuer is the one often paying for frauds. They want to minimize card numbers to be stolen and if stolen they should be blocked.

In this master thesis paper I investigate internet payments, what security requirements there are for them, what threats there are and how to mitigate them.

I analyze a payment server built for in–game payments using a method that after some modification can be integrated with the development cycle. The method uses the public library Common Weakness Enumeration (CWE).

Sammanfattning

Säkra internetbetalningar och en studie

Internetbetalningar ökar. Folk köper inte bara fysiska varor på internet utan också lotterier, biljetter och andra virtuella varor så som virtuella pengar. Det blir en allt viktigare del av ekonomin.


I detta examensarbete undersöker jag internetbetalningar. Jag tittar på säkerhetskraven för denna typ av lösningar, vilka hot det finns och hur de kan minskas.

# Contents

## Abbreviations and Acronyms

1. Introduction ............................................................................................................. 1  
   1.1 Problem Description ....................................................................................... 1  
   1.2 Restrictions ..................................................................................................... 2  

2. Payment Systems .................................................................................................... 3  
   2.1 Payments in Games ....................................................................................... 3  
   2.2 Payment with Credit Card ............................................................................ 3  
   2.3 Payments with Payment Service Provider .................................................. 5  

3. Solutions and Security Issues ................................................................................ 7  
   3.1 Security Requirements .................................................................................. 7  
   3.2 Threats and Solutions ................................................................................... 8  
   3.3 How to Measure Security ............................................................................ 11  
   3.4 Security Standards ....................................................................................... 16  

4. Methodology .......................................................................................................... 18  

5. Case Study .............................................................................................................. 22  
   5.1 The System ..................................................................................................... 22  
   5.2 Step 1: Functional Requirements .................................................................. 25  
   5.3 Step 2: Security Goals or Assets .................................................................. 25  
   5.4 Step 3: Security Requirements ..................................................................... 26  
   5.5 Step 4: Construct Outer Arguments ............................................................. 26  
   5.6 Step 5: Identify Risks ................................................................................... 28  
   5.7 Step 6: Classify Risks .................................................................................. 35  
   5.8 Step 8: Verify Mitigation ............................................................................. 36  
   5.9 Step 9: Prioritize Risks ................................................................................. 44  
   5.10 Fraud Prevention ......................................................................................... 46  
   5.11 Results .......................................................................................................... 51  

6. Conclusions ............................................................................................................. 53  

7. Discussion .............................................................................................................. 54  

Bibliography .............................................................................................................. 56
Abbreviations and Acronyms

CA – Certificate Authority
CAPEC – Common Attack Pattern Enumeration and Classification
CERT – Computer Emergency Response Team
CSI – Computer Security Institute
CVSS – Common Vulnerability Scoring System
CWSS – Common Weakness Scoring System
CWE – Common Weakness Enumeration
CWRAF – Common Weakness Risk Analysis Framework
DDoS – Distributed Denial of Service
NIST – National Institute of Standards and Technology
OWASP – Open Web Application Security Project
PCI – Payment Card Industry
PCI-DSS – Payment Card Industry Data Security Standard
PSP – Payment Service Provider
SANS Institute – private U.S. company that specializes in information security and cybersecurity training
SOA – Service Oriented Architecture
SOAP – Simple Object Access Protocol
1 Introduction

Computer security is an old topic and there is a lot of literature about it. E-commerce adds another perspective to computer security. Abdellaoui, Pasquet, and Berthelier writes about the security requirements for this kind of applications [4] such as integrity, confidentiality and privacy. Laudon and Traver writes about the security threats and solutions [6]. Internet is always evolving and there is need for a way to manage security in this dynamic environment. Common Weakness Enumeration (CWE) is a database of weaknesses commonly found in systems that is continuously updated reflecting the changing environment [15]. CWE considers security threats. Franqueira, Tun, Yu, Wieringa, and Nuseibeh proposed a method to find security requirements that takes into account the changing security conditions when creating a new application [22]. The method is for showing that the application meets the security requirements by argument, but it is not for verifying that the application meets the requirements. I modified the method to be used for verifying an already created application. I demonstrate the modified method in a case study of a payment server. The main contribution of this paper is using the modified method to find and verify security requirements of a typical payment server implementation.

1.1 Problem Description

What security requirements are there for this kind of applications? What threats are there and how are they mitigated? I investigate what security requirements and threats are for this kind of applications in general. I also write about how to mitigate the security threats.

After finding the security requirements and threats, how can we make sure that an application has a good enough security? I look into how the security of a system can be measured.

What help is there for companies and organizations to maintain a certain level of security and reliability? I look into what security standards and public catalogues there are.

I review a new payment service for in–game payments from a security perspective. The security requirements for the payment service are found. I also verify that all security requirements are full filled. I use a risk based argumentation method for doing the analysis of the payment system. After analyzing the system I verify it by doing code review, information flow analysis and penetration tests.
The support organization does not have the proper tools to check that the transactions are correct to prevent fraud. I investigate the historic transaction data to see if there is a way to identify and block fraudulent players.

1.2 Restrictions
There are many functions in a payment service. It has an administrator user interface, report functionality both for customers and for finance and reconciliation functionality. I will restrict the analysis to the actual payment function.

I will not test denial of service attacks as it if successful can make the game site unstable and the company can lose money as payments cannot be done.
2 Payment Systems

This chapter describes how payments are handled. The first part considers payments in games. The second part considers credit card payments in general. The third part is about payments with a service provider.

2.1 Payments in Games

How are payments in games handled? I will describe payment systems in three games.

In the game Star Doll the player can buy star dollars. The player is offered different payment methods based on the currency, country and amount. For credit card payments the player enters sensitive information on the site but the payment is handled by a payment service provider in the background. If the player selects any other payment method the player is redirected to the payment service provider for sensitive data entry. The company handles sensitive credit card data but the payment transaction is delegated to a payment service provider.

The game Candy Crush and other games in King.com work in a similar way but they do not handle credit card data in plain text. The player is also redirected to a payment service provider for payment in the King.com site. A payment method for “one click payments” is automatically set up for the player. Next time the player wants to buy something the player has a fast track payment and does not need to enter credit card data again.

The game World of Warcraft has an in-game store where players can buy pets and mounts. The player has to have set up a payment method to be able to buy from the game. The payment method is set up on their site in the browser. For credit card payments the player enters sensitive information on the site but if the user selected PayPal he/she is redirected to PayPal for sensitive data entry.

Payments in games use the same mechanism as payments in other merchant sites. The difference is the goods. The payment solutions for games have the same issues as the payment solutions for other merchants. These issues are discussed in chapter 3.

2.2 Payment with Credit Card

There are many steps when making an online transaction with credit cards. There are many actors involved. Banks and institutions show a somewhat
similar picture of the flow. Here are the payment flows from Visa, Mastercard and Canadian Bank Association (CBA).

The five steps in the transaction flow can be seen in Figure 2.1 from Canadian Bank Association (CBA).

![Figure 2.1: Credit card transaction flow from CBA [1].](image)

Figure 2.2 shows the transaction flow from visa. It does not show the payment processor.

![Figure 2.2: Credit card transaction flow from Visa [2].](image)

Figure 2.3 shows the transaction flow from master card. The payment processor is not shown on this diagram either.

![Figure 2.3: Credit card transaction flow from Master Card [3].](image)

The flow including the payment processor:

1. Buyer presents the card details
2. Merchant sends transaction details to payment processor
3. Payment processor or acquirer sends the transaction to the credit card company’s network
4. Card company sends the transaction to card issuer for authorization
5. Issuer approves the transaction
A merchant is the one selling the wares to the customer. It can be a company selling physical goods such as CDs, food or shoes. It can be a company selling virtual goods such as music, games or game items.

The payment processor allows the merchants to accept different cards and redirects to the correct credit card network. An example is Global Payments. [1]

The acquiring bank is the bank or financial institution where the merchant has an agreement for accepting money from card transactions. They also act as intermediary if a customer should have a dispute about a payment. [2] Examples of acquirers are Babs, Euroline and Teller.

Credit card companies such as Mastercard and Visa operate international networks that process card transactions. [1]

The issuer is a bank or other financial institution. The issuer is responsible for card security and is the one that often takes the costs when security is compromised. The issuer bank is associated with credit card brands such as Visa, Mastercard etc. [1]

There are companies that offer merchants easy access to the payment service. They are called Payment Service Providers (PSP). They offer connection to multiple acquirers, multiple payment methods and other services. Merchants can choose to buy the whole technical solution for payments or only parts of it such as transaction processing. [4] Examples of PSPs are WorldPay and Payzone.

Many merchants use PSPs for their online payment. In this way they can choose not to handle card numbers in plain text in their systems. They can also choose to handle card number in plain text but the transaction is handled by the PSP.

### 2.3 Payments with Payment Service Provider

There are two ways for a merchant to handle payments. The traditional solution is that the merchant handles the connection to the acquirer themselves. This is an expensive solution. The merchant is responsible for the security which leads to huge costs and risks. The other approach is to integrate with a PSP. The merchant is responsible for the safe connection to the PSP, but it is the PSP that is responsible for the payment transaction. The merchant does not need to handle card numbers at all. He just redirects the buyer to the PSP for the entry of card details and the buyer gets redirected back to the merchant when the payment is done. [4]
The reason why merchants would have their own more expensive integration to their acquirer is that the PSP charges a rate per transaction. When the number of transactions is very high it might be more profitable to have their own connection to the acquirer.
3 Solutions and Security Issues

This section deals with security issues. What security requirements are there on this kind of applications? I explore what threats there are and what countermeasures there are to mitigate the threats. Finally I address the difficulty of measuring security and mention some standards set to guarantee a minimal security.

3.1 Security Requirements

The requirements a payment system has to satisfy can be grouped in technical, economic, social and legal requirements. [4]

Table 3-1: E–payment system requirements [4].

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technological</td>
<td>- Authentication</td>
</tr>
<tr>
<td>Requirements</td>
<td>- Integrity</td>
</tr>
<tr>
<td></td>
<td>- Confidentiality</td>
</tr>
<tr>
<td></td>
<td>- Non Repudiation</td>
</tr>
<tr>
<td>Economic Requirements</td>
<td>- The cost of Transaction</td>
</tr>
<tr>
<td></td>
<td>- Atomic exchange</td>
</tr>
<tr>
<td></td>
<td>- User Range (System accessibility)</td>
</tr>
<tr>
<td></td>
<td>- Value Mobility (value is used anywhere)</td>
</tr>
<tr>
<td></td>
<td>- Financial Risk</td>
</tr>
<tr>
<td></td>
<td>- Return On Investment (ROI)</td>
</tr>
<tr>
<td>Social Requirements</td>
<td>- Privacy</td>
</tr>
<tr>
<td></td>
<td>- Accessibility</td>
</tr>
<tr>
<td></td>
<td>- Mobility</td>
</tr>
<tr>
<td>Legal Requirements</td>
<td>- Digital Signature</td>
</tr>
<tr>
<td></td>
<td>- Digital Transfer</td>
</tr>
<tr>
<td></td>
<td>- Legality of Payment</td>
</tr>
<tr>
<td></td>
<td>- E-commerce Contracts</td>
</tr>
<tr>
<td></td>
<td>- Technical Standards</td>
</tr>
<tr>
<td></td>
<td>- Rental Taxes</td>
</tr>
<tr>
<td></td>
<td>- International Transactions</td>
</tr>
<tr>
<td></td>
<td>- Intellectual Property Protection</td>
</tr>
</tbody>
</table>

Not all the requirements in Table 3-1 above concern security. The security related requirements are authentication, integrity, confidentiality, non-repudiation of transaction, privacy and accessibility. The legal requirements are regulations put to guarantee the other requirements i.e. digital signature is to guarantee integrity and technical standards such as PCI–DSS is to prevent fraud and security breaches.

Authentication is fundamental in a security context. It is defined as “The process of verifying an identity claimed by or for a system entity…” in [5].
Integrity assures that information is changed in an authorized manner and that systems and programs are not manipulated unauthorized.

Confidentiality and privacy assures that sensitive information is only made available to authorized users.

Nonrepudiation provides protection against a user denying receiving or sending a message.

Availability and accessibility assures that the system is available to authorized users [5].

3.2 Threats and Solutions
Sensitive information resides in computers and servers that are not completely secure. Sensitive information has to be sent on unsecure media. There are many threats to client–server solutions as can be seen in Figure 3.1.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure3_1.jpg}
\caption{Vulnerable points in an e–commerce transaction [6].}
\end{figure}

In both the client side and the servers there can be malicious code such as viruses, worms and Trojan horses. On the server side companies tend to be more careful and use anti–virus programs but customers are in general not that rigorous about it. The impact on a server can be disastrous resulting in the service being down, loss of data or data theft. On the client side spyware can be used to steal confidential information. Another method to steal con-
Confidential information from customers is phishing. It is a way of tricking a user to enter a site pretending to be another site. The user then enters information thinking it is the right site and it is stolen.[6]

Hacker attacks have diversified over time and becoming more malicious, stealing goods and information. There are some cases where millions of credit card numbers have been stolen. Companies protect themselves by contracting benign hacker groups that try to find weak spots on the system.[6]

Hackers also do Distributed Denial of Service (DDoS) attacks for various reasons. Some blackmail companies, threatening doing an attack. Others do it for fun and there are some that are politically motivated. It is not possible to protect completely against DDoS attacks. Many times they use spoofed IP–addresses when doing an attack. The internet service providers can filter out the packages with spoofed IP–addresses but many providers do not do that because it impacts performance. To protect against overflow in the TCP connection table there is a modified version of TCP connection handling code. The disadvantage of this is that it requires more computation and is incompatible with some extensions of TCP. It is not enabled by default and it is used when the TCP connection table overflows. Another way is to simply drop a connection from the table when it overflows. [5]

It might be difficult to identify an illegitimate request from a legitimate one on an application level. One way is using a captcha for resource–intensive requests. The organization has to have a plan in case an attack should occur. The network should be monitored and there should be an automated intrusion detection system running. When an abnormal network traffic is detected the first step is to analyze the packets with some tools to find a pattern and then to filter them out. There should be backup servers to switch to if it is not possible to filter them out. [5]

Anytime the information goes through a network or internet it can be subject to sniffing. Sniffing is eavesdropping the network with a program. It is legal for the authorities to do it in some countries such as the US. [6]

Privacy is threatened by sniffers (network monitor programs). The transaction should be encrypted and should not reside in intermediate nodes more than necessary to ensure confidentiality. Error checking codes helps to verify that the data has not been modified, to ensure integrity. The weakness of encryption techniques using secret keys, public keys and private keys is the key handling. [7]

Encryption is effective to prevent sniffing. There is symmetric encryption where both sides share one secret key used both for encryption and decryption. It is an efficient method. The issue using only this method is the key
distribution. To take advantage of the efficiency of the method and solving the key distribution it is combined with asymmetric encryption. Asymmetric encryption is the private–public key encryption where each actor has a public key and a private one. The sender encrypts its message with the public key of the receiver. The message can only be decrypted by the receiver with its private key. An encrypted message with a public key can only be decrypted with the corresponding private key. This algorithm is slow. Both symmetric and asymmetric encryptions are combined in SSL to create an effective and efficient method. A secret used to generate the secret key is exchanged during the handshake using the other one’s public key. The secret key is then used to encrypt and decrypt the rest of the messages during the session. Public keys are managed by Certificate authorities (CA). A CA is a trusted third party that signs the public key together with the ID into a certificate. The certificate is then used to authenticate the owner of the public key. [5]

The insiders are also a threat. Insiders have privilege access to the systems. The Computer Security Institute (CSI) report for 2009 showed that insider abuses were second most frequent type of attacks. Insiders can also do damage either intentionally or unintentionally by negligence or ignorance. [6]

There are also bugs in the software (operating systems, browsers, servers, etc.) that can make the system vulnerable. The bugs are corrected and patches are released but there is a time lapse between the discovery of vulnerability and the release of a patch and the patch being installed. The vulnerability can be exploited during this time. [6]

There are various ideas to prevent fraud when card data is stolen. Most use machine learning methods with artificial intelligence or Bayesian learning. Others use data mining systems and pattern matching. [8]

One interesting solution is to use hidden Markov model to identify deviation in the user behavior. The transaction amount is used for differentiation. The hidden Markov model is a probabilistic model used for among other things speech recognition. [9]

Yan & Chiu proposes a confirmation system Notified Credit Card Payment System (NCCPS) that uses SMS. They propose using the Web Service Architecture for this. The NCCPS acts as PSP for the merchant. NCCPS receives the payment transaction from the merchant. Having the card number NCCPS asks the bank for the phone number to the card holder. A SMS confirmation is sent to the card holder. The transaction is sent to the bank for completion after the customer replies to the SMS. If the card holder is not the one making a payment he/she can call a customer service and the transaction is rejected and other actions can be taken. The customer service is also connected to NCCPS calling web services. [10]
How to Measure Security

How do you know how much security your solution has? How can you compare the security between releases? There are many methods and frameworks for measuring security, both corporate and community developed. I will mention some methods to measure security and describe two of them in more detail.

CERT is a nonprofit federally funded organization sponsored by the U.S. Department of Defense. CERT does lot of research on security risk analysis and metrics and have developed the integrated measurement and analysis framework (IMAF). It is a framework for measurements in complex environments. [11]

Microsoft has also a proprietary scoring system. It is based on customer feedback and it is primarily intended for customer to decide whether to upgrade their software. [12]

Next I am going to describe CVSS and CWSS. Both of them are open frameworks and very similar.

3.2.1 Common Vulnerability Scoring System

The common vulnerability scoring system (CVSS) is an open framework to grade vulnerabilities. It is developed by the forum of incident response and security teams (FIRST).

It has three groups with metrics as can be seen in Figure 3.2.

![Figure 3.2: Metric groups in CVSS](image)

Figure 3.2: Metric groups in CVSS [13].
The base metric group contains fundamental characteristics of vulnerabilities such as authentication. The temporal metric group contains metrics that change over time such as exploitability. The third group contains metrics that are environmental specific such as target distribution.

Each metric in the base group is valued. A score is then calculated for the group by a complex formula. It should result in a value with one decimal between 0 and 10.

The score can be refined with the values from the temporal metrics group. A new score is calculated with another complex formula using the score from the basic group with values from the temporal metrics group. The score calculated cannot be higher than the basic score and maximum 33 % less than the base score.

The score can be further refined with the values from the environmental metrics group. A new score is calculated with another complex formula that uses the calculated score and values from the environmental group. This calculation will produce a score that is less than or equal to the score from the previous calculation. [13]

3.2.2 Common Weakness Scoring System

The common weakness scoring system (CWSS) is a community based system to score weaknesses in a system.

The system combines sixteen factors that are graded and weighted to calculate a score.

The factors are arranged in three groups (See Figure 3.3): base finding, attack surface and environmental.
Figure 3.3: Metric groups in CWSS [14].

When a vulnerability is found it is checked against each factor. Each factor is given a value and each value has a weight. Table 3-2 shows values and weights for technical impact.
Table 3.2: Weights for technical impact.

<table>
<thead>
<tr>
<th>Value</th>
<th>Code</th>
<th>Weight</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Critical</td>
<td>C</td>
<td>1.0</td>
<td>Complete control over the software being analyzed, to the point where operations cannot take place.</td>
</tr>
<tr>
<td>High</td>
<td>H</td>
<td>0.9</td>
<td>Significant control over the software being analyzed, or access to critical information can be obtained.</td>
</tr>
<tr>
<td>Medium</td>
<td>M</td>
<td>0.6</td>
<td>Moderate control over the software being analyzed, or access to moderately important information can be obtained.</td>
</tr>
<tr>
<td>Low</td>
<td>L</td>
<td>0.3</td>
<td>Minimal control over the software being analyzed or only access to relatively unimportant information can be obtained.</td>
</tr>
<tr>
<td>None</td>
<td>N</td>
<td>0.0</td>
<td>There is no technical impact to the software being analyzed at all. In other words, this does not lead to a vulnerability.</td>
</tr>
<tr>
<td>Default</td>
<td>D</td>
<td>0.6</td>
<td>The Default weight is the median of the weights for Critical, High, Medium, Low, and None.</td>
</tr>
<tr>
<td>Unknown</td>
<td>UK</td>
<td>0.5</td>
<td>There is not enough information to provide a value for this factor. Further analysis may be necessary. In the future, a different value might be chosen, which could affect the score.</td>
</tr>
<tr>
<td>Not Applicable</td>
<td>NA</td>
<td>1.0</td>
<td>This factor is being intentionally ignored in the score calculation because it is not relevant to how the scorer prioritizes weaknesses. This factor might not be applicable in an environment with high assurance requirements; the user might want to investigate every weakness finding of interest, regardless of confidence.</td>
</tr>
<tr>
<td>Quantified</td>
<td>Q</td>
<td></td>
<td>This factor could be quantified with custom weights.</td>
</tr>
</tbody>
</table>

The last value “Quantified” is for customizing the weight. CWRAF, that I will mention later, can be used for calculating a weight that considers the business value.

Each group has a formula for calculating a score for the group. The scores for each group are multiplied to one single score. The score is a measure of the vulnerability that should be used for prioritizing and deciding whether the vulnerability is to be fixed. [14]

3.2.3 Common Weakness Enumeration

The common weakness enumeration (CWE) is a list of weaknesses found in software. It works as a common categorized list with software weaknesses that are described in a uniform way.

The weakness record contains among other things a description, consequences with technical impact, likelihood of exploit, examples and mitigations. Even if there are more than 600 CWE the technical impact is summarized to eight different ones: read data, modify data, execute unauthorized
code or commands, gain privileges / assume identity, bypass protection mechanism, hide activities, DoS: resource consumption and DoS: unreliable execution.

The list is formalized so that scanning tools can use the list when looking for weaknesses.

It is maintained by a community of big companies and organizations. [15]

3.2.4 Common Weakness Risk Analysis Framework

The common weakness risk analysis framework (CWRAF) helps software developers prioritize weaknesses based on their business and deployed technologies. Users can generate a CWE top list for their business. [16]

To use CWRAF the user has to describe the business and business environment. A user can use already created vignette or creates a new vignette. A vignette is a formal description of the business environment. It has a list of archetypes. An archetype is a technical capability or architecture i.e. database, web browser, service–oriented architecture etc. There is a list of archetypes. A vignette has a business value context and technical impact scorecards. A business value context describes in a general way the security relevant assets and interfaces. It also prioritizes the business domain if an attack is successful. An example of a business value context is “Security incidents might have organizational impacts including financial loss, legal liability...”. Technical impact scorecards give a score for each combination of technical impact and layer. The technical impacts are the same as the ones for CWE. There are four layers: Enterprise, Application, System and Network.

Having a vignette one can use it to create a top list of CWE for the specific business and technology but also for scoring discovered weaknesses in the system. [16]

The following steps have to be taken to calculate a score for a weakness:

1. Find the CWE entry for the weakness and get its potential technical impacts.
2. For each technical impact of the CWE find the importance rating of the corresponding technical impact in the vignette.
3. The score is the biggest importance rating divided by 10.

There are some manual steps and some automatic steps to create a top list:

1. Select the relevant CWE or a sub set of them based on the archetypes (technology).
2. Match the technical impact of the vignette with the ones of the CWE.
3. Calculate the CWSS score for CWE based on vignette.
4. Rank the CWE to make a top list.

I think adding the aspect of deployed technology and business importance when prioritizing the CWE is important to get a more adjusted grade and I will use this method to prioritize the weaknesses found when doing the analysis of the payment system in chapter 5.

### 3.3 Security Standards

What is good enough? This is a management question. Not having enough security is taking a risk and security costs money. It is a matter of cost vs risk. There are security standards to enforce a minimum security for organizations and companies handling sensitive information. There are also top lists to help companies prioritize.

The payment card industry (PCI) security standard council has developed the security standard data security standard (PCI–DSS) that companies handling card numbers have to follow. The main purpose is to protect card information. The standard is extensive and covers both physical security and software security during development phase and in production environment. The companies that handles encrypted card numbers only need to fill out a self-assessment questionnaire but companies that handle decrypted card numbers have to do a more extensive on-site assessment. This is done by a qualified security assessor and approved scanning vendors that are very expensive. [17]

The national institute of standards and technology (NIST) in the US has published a list of security controls for federal organizations. The controls are countermeasures to satisfy the security requirements defined in FIPS Publication 200 “Minimum Security Requirements for Federal Information and Information Systems”. The organizations categorize themselves with help of FIPS Publication 199 “Standards for Security Categorization of Federal Information and Information Systems”. Using the category the organizations can find the security requirements for them. Having the requirements they select the controls they have to implement. The controls have a priority and are assigned to baselines. There are three baselines: low, moderate and high. There is a huge number of controls. They are organized in eighteen families or topics. A family can be i.e. access control, audit and accountability and program management. The controls include management tasks such as access control policy as well as developer tasks such as details about audit logging. [18]

The controls published by NIST are so extensive that SANS institute selected a sub set of them as the most critical ones that are most effective. It is the
20 Critical Security Controls (CSC). They also describe how to implement it and how to measure it. [19]

SANS institute also selects the 25 most dangerous software errors among the over 600 listed weaknesses in CWE. They are using the scoring system CWSS to make the selection. They prioritize them according to weakness prevalence, importance, and likelihood of exploit. It is intended as a tool for education and for programmers. [20]

The open web application security project (OWASP) is an open community. They rank the top 10 security issues among thousands of weaknesses reported by companies. They prioritize them according to prevalence, exploitability, detectability and impact estimates. [21]
4 Methodology

The methodology for making this paper is described in this chapter.

I interviewed some people working in game companies about how the payments in games are handled as I did not find any useful papers on the topic.

I searched banks and credit card sites for information about credit card payments in general to understand involved actors and the flow of information.

I did literature search on internet payments, security requirements and security issues for this kind of applications and found some papers describing different solutions that addresses the issues.

I needed a way to asses and quantify security and found an existing methodology that I could expand to include "verification of mitigation" described below.

To find possible vulnerabilities I did code review and analyzed the information flow. We tested the vulnerabilities that I found together with the development team. The development team fixed the vulnerabilities.

At the end I made some recommendations for integrating a security methodology with the development cycle.

The methodology for making the analysis is based on a method proposed by Franqueira, Tun, Yu, Wieringa and Nuseibeh [22]. The method describes system behavior in order to find vulnerabilities and threats using public catalogues such as CWE. The method is iterative and its goal is to improve the security to a good–enough level. There are the eight steps (See Figure 4.1). I modified this method to suit the needs for the actual situation.
The unmodified method steps:

1. Identify functional requirements.
2. Identify security goals, assets to be protected.
3. Identify security requirements. These requirements are constraints on the functional requirements in order to protect the assets.
4. Construct outer arguments. Outer arguments show that the specification of the application and domain premises entails the security requirements. It is normally written in propositional logic. Example: The transaction is encrypted \(\rightarrow\) Transaction shall remain confidential
5. Identify risks. Questions the premises from outer arguments. Public catalogs such as the common weakness enumeration (CWE) and the common attack pattern enumeration and classification (CAPEC), are searched for known weaknesses regarding the premise.
6. Classify risks. There are two classes of risks. There are risks that the system can mitigate and there are risks the system cannot do anything about. They are responsibility of the system context. The functional requirements can be altered because there can be a change of obligations from the system context to the system. That is the reason for the arrow in Figure 4.1 from step 6 to step 1.
7. Mitigate risks. Search in the public catalogue, for instance CWE, how to mitigate the identified risks that are classified as belonging to the system. Some mitigation might introduce new risks.
8. Prioritize risks. Risks are prioritized according to their severity in the catalogue.
The method is recursive. For each mitigation in step 7 one can potentially jump back to step 5 as can be seen in Figure 4.1.

The method is for identification and mitigation of threats but not made for verification. With verification I mean to verify that the mitigation is implemented correctly by doing code review or penetration tests depending on the detection method for the CWE. I modified this method by adding a verification step (See Figure 4.2). The verification step is done after the mitigation. If the verification fails the mitigation has to be redone. I verified that the risks are mitigated by finding weak spots, doing tests and code review. Detection methods for the CWEs can be found on the public CWE catalogue.

Instead of searching the public catalogs CWE with more than 600 weaknesses I restricted the search to CWE top 25 and OWASP top 10. Instead of searching CAPEC I looked for the attacker model for this specific case, possible attackers and their capabilities. I did this to reduce the scope of the work.

Step 7 is to mitigate the risks. It was done by the development team. The last step is to prioritize the risks. For prioritizing the risks I used the common weakness risk analysis framework (CWRAF) in combination with the common weakness scoring system (CWSS). This way the prioritizing takes into account business domain related and technological issues. I chose CWSS instead of CVSS because it can be used in combination with CWRAF, it can handle unknown values and because it is used for prioritizing the SANS/CWE top 25 list. The authors of the method propose using CWSS. More on the differences between them can be read on cwe.mitre.org appendix A.
Figure 4.2: Modified method that I used.

At the end I looked at fraud prevention. Fraud is also a risk. It is highly prioritized by the company. The credibility of the company would diminish if players were able to cheat the system. I did literature search on detection of fraudulent card transactions and found some interesting papers on hidden Markov model but I could not use that method. I used the attacker model and historic transactions to identify fraudulent behavior.
5 Case Study

The case is a security analysis of a payment server used by games so that players can buy virtual goods while inside the game.

The company has already a security department that regularly performs black–box penetration testing and sometimes code reviews on the APIs. They hire a big international company for the black–box testing that is specialized and expert in security. The security company performs tests on the whole site but not on the payment server in specific. The game company also suspects fraud from players and want a report for detecting fraudulent players.

I will first describe the system and the system behavior in section 5.1, then I go through all the steps from the methodology as explained in Section 4. The verification in section 5.8 is done by code review and penetration testing.

Then I look for patterns in fraudulent behavior. Unfortunately there was not enough time to do some more advanced statistical analysis other than some simple heuristic methods.

5.1 The System

There are four actors involved in the payment process (See Figure 5.1). The player’s browser interacts with the player. The game server runs the game and calls the payment server to initiate a payment. The payment server communicates with the Payment Service Provider (PSP) and redirects the player browser to the PSP site. The PSP takes care of the actual payment. They communicate with institutions, banks and operators etc.

The game server initiates the paying process. The payment server communicates directly with the player’s browser during the paying process.
5.1.1 Payment flow

There are some steps done before the actual payment flow starts that are good to know for the understanding of the analysis. When the player selects to pay the system retrieves a list of payment methods based on country, currency and amount. The actual payment flow starts when the user choses payment method. A payment method is supported by a PSP. The integrated PSPs are ACapture, Worldpay, Payvision, Paysafecard, Boku, ZONG and Paypal. ACapture and Worldpay are big PSPs having a lot of local payment methods for different countries, also offering currency exchange. Boku is a French PSP charging the carrier bill. Zong is also a PSP charging the carrier bill. Paysafecard have their own card that a customer loads with money.

The payment flow differs slightly for different PSP but it looks in general like Figure 5.2. The client application and the payment server are within the company.
The main steps in the flow are:

1. The payment server calls the PSP to initiate a payment. Some PSP create a transaction at this step and some PSP create a session. A transaction identifier or a session token is returned. The transaction is valid from 6 min to 5 days depending on the PSP. There is one PSP that skips this step all together.
2. The player’s browser is redirected to the PSP for entry of sensible customer data. Some PSP handles card payments, others do transaction directly from the bank account, other charge the internet bill and others charge the phone bill.
3. The PSP notifies the payment server that the payment is ready. For some PSP this is the end of story but other PSP are not finished. The payment server has to do an additional call for the payment to actually take place.

The communication with external systems are potentially weak spots. It can be subject to sniffing or alteration of message described in section 3.2. If the
amount or currency is altered the player might be paying less. If the transaction id or the result of the payment is altered a purchase with a declined transaction might be approved. There are six potentially weak spots in the flow. They are marked with stars. The first one is the communication with the game server. The amount can potentially be changed. The second weak spot is the call for initiation of the transaction. The amount could potentially be changed. The third weak spot is the redirect to the PSP for entry of sensitive data. The amount and transaction id can potentially be altered leading to paying less for the wares. The fourth weak spot is when the customer is redirected back to the payment server. The payment result can be altered or replaced with the result for another transaction. The fifth weak spot is the callback from the PSP to the payment server. This callback might be faked by an impostor leading to a transaction being incorrectly approved. The sixth one is when the result is communicated back to the game server. I will discuss this further when doing the code review in section 5.8.1.

5.2 Step 1: Functional Requirements
The first step of the methodology is to find functional requirements.

The system has many functional requirements but I will only analyze the requirements for doing the payment. There are other requirements for reconciliation, reporting, for administration of the payment service such as enabling and disabling Payment Service Providers (PSP), for the player to see payment history and remove a fast track payment method.

The functional requirements for doing a payment are:

FR1: The player chooses to pay with either credit/debit card or another payment method supported for its country, currency and amount. The player is redirected to the PSP for the selected payment method for entering sensitive information. The player is redirected back to the payment server and the correct result is displayed.

FR2: The player chooses to pay with an already stored payment method. The player does not enter card details. It is already stored. This payment method is called fast track.

5.3 Step 2: Security Goals or Assets
The second step of the methodology is to identify the security goals or assets to be protected.

The payment server does not handle card numbers in plain text. The sensitive payment information is handled by the PSPs.
The most important information to protect is the transaction. It is essential that it is not altered. If the amount or currency is changed the player pays less for the wares.

The transaction result should not be altered in transit. If a decline result is changed to an approved the player gets the wares for free. If an approved result is altered with another transaction’s identification then the player gets the wares for free.

The billing agreement (fast track information) is stored in the payment server. It has to be protected. If someone acquires a billing agreement they might use it to pay with a card belonging to somebody else without entering card details.

The system should not go down. The company loses money for every minute the system is down.

5.4 Step 3: Security Requirements

The third step of the methodology is to identify the security requirements. The security requirements are restrictions on the functional requirements in order to protect the identified assets. They have to be defined on a high level.

SR1: The integrity of the transaction has to be secured all the way from the payment method selection until the transaction is received by the PSP.

SR2: The integrity of the transaction result has to be secured from the time the PSP sends it to the payment server until the payment server receives it.

SR3: The confidentiality of the billing agreement has to be secured at all times. It should not be possible for one player to use the billing agreement of another player.

SR4: The system has to be available.

More about integrity, confidentiality and availability can be read in section 3.1.

5.5 Step 4: Construct Outer Arguments

The step to construct outer arguments is to make sure the specification of the system and system context entails the identified security requirements. The outer arguments are documented in propositional logic. Premises are found in specification of the system and system context. If the premises are true then the conclusions are also true. The conclusions are the security requirements.
There is no formal written specification of the system other than the code and some diagrams. There is no deployment documentation describing the environment. I have to find the information by asking developers and other workers.

I have identified the following domain objects: Player browser, Game server, Payment server, PSP.

Security Functions

SF1: The communication between game server and payment server is handled with JavaScript Object Notation Remote Procedure Call (json–rpc) calls from within the internal local network. The access to the internal local network is protected from unauthorized access reducing the risk of sniffing.

SF2: The communication between PSPs and payment server is either with HTTPS or having a checksum. This is a protection against alteration of message and sniffing described in section 3.2.

SF3: The payment server is deployed in two physical machines with a load balancer. This is to ensure availability described in section 3.1. One of them serves as a backup server to reduce the damage of a DDoS attack described in section 3.2.

Premises

P1: The payment server API receiving transaction data from the game server is only receiving the transaction data as json–rpc calls from within the local network → the transaction is sent from the payment server to the PSP with HTTPS and is therefore encrypted.

P2: The transaction is sent from the payment server to the PSP with HTTPS and is therefore encrypted → PSP returns a session id with HTTPS.

P3: PSP returns a session id with HTTPS → the payment server sends a redirect URL to the player browser.

P4: PSP returns the transaction result with HTTPS → the payment server returns the transaction result to the game server doing a json–rpc callback.

P5: The PSPs that do not return the transaction result with HTTPS have a checksum → the payment server returns the transaction result to the game server doing a json–rpc callback.

P6: The billing agreement is stored encrypted in the database.

Triggering Assumption
A1: Player browser send the transaction data to the game server → Game server sends the transaction data to the payment server in a json-rpc call from within the local network.

**Conclusion**

P1, P2, P3 \(\models\) C1: integrity and confidentiality of the transaction

C1, P4 or P5 \(\models\) C2: integrity of the transaction result

P6 \(\models\) C3: confidentiality of the billing agreement

Assuming the premises the security requirements are fulfilled and also the functional requirements.

Next step is to challenge the premises to find the risks.

**5.6 Step 5: Identify Risks**

This step is to challenge the found premises. To identify the risks I will first look at the attacker model. I will find out who the possible attackers are and what capabilities they have. Then I will go through CWE/SANS top 25 and OWASP top 10 to identify which ones apply to the system and challenge the premises.

**5.6.1 Attackers**

There are many possible attackers to a famous company. There are registered players that want to get more in-game money without paying for it. There are elite hackers that do it for the thrill and fame. There are criminals using stolen cards or criminals using other player accounts. There are also possible insider attackers.

Most registered players are innocent. They just want to play as a distraction. The registered players in general do not have hacker skills. Some of them utilize the possibility to buy virtual money, spend it and then do a refund. They just have to go to the bank to be reimbursed. It is very easy for the customer and it is costly for the company. There is a fee for each refund. The company can get blocked if the percentage of refunds exceeds a certain limit.

Then there are some curious ones, script kiddies, which look at the source of the pages and how the messages are sent. They have access to tools for doing man in the middle attacks. Script kiddies do not write their own tools or modify existing tools. Tools are very easy to find and to use. Script kiddies read forums and keep themselves informed but are in general not highly
skilled. They cannot do much damage to a well-protected site [23]. The attack from this group is limited to their own accounts.

The hacktivists or elite hackers are the most persistent. Fame is the main motivator for this group. They have deep knowledge of the underlying technology and access to powerful tools. Elite hackers have the skill to make their own tools if necessary and are organized in groups [23].

One of these groups called Derp has specialized in attacking gaming sites. They mostly do DDoS attacks and have followers that help them in the DDoS attacks by giving them access to their computer but also use hijacked computers infested with malware. (http://www.theguardian.com/technology/2014/aug/28/derp-inside-hacking-group-cyber-attacks-phantomlord)

Because there can be many default users and passwords in internal systems it might be possible for employees to use them to cause harm or gain access to confidential information. 70% of the security breaches are insider attacks according to the FBI. It can be curiosity, revenge or espionage that drives these people. A skilled and motivated insider attacker is dangerous. The attacker can redirect the users to a malicious site or simply shut down the system by doing sabotage. [24]

5.6.2 CWE/SANS Top 25

There are more than 600 CWE. I order to restrict the scope of the work but still get the best out of it I will only look at the top 25 CWE. If working with 600 CWE you start with the premises and look for CWE challenging the premises. I will instead go through the CWE top 25 list and find the ones that challenge the premises. I will go through the list and see what vulnerabilities can be ruled out and what vulnerabilities challenge the premises. Only the CWE that challenges the premises are kept for next step. The others are not applicable to the payment system.

In Table 5-1 I will shortly describe each CWE in the top 25 list and explain why some are ruled out. For more detail information about the CWE see [15].
<table>
<thead>
<tr>
<th>ID</th>
<th>Name and Description</th>
<th>Premises Challenged</th>
</tr>
</thead>
<tbody>
<tr>
<td>CWE–89</td>
<td>Improper Neutralization of Special Elements used in an SQL Command (“SQL Injection”)</td>
<td>Premise P6 is challenged because the transaction data and billing agreement are stored in the DB.</td>
</tr>
<tr>
<td></td>
<td>This is exploited when code is sent to an interpreter that is expecting to receive data. The code can be sent in a HTML form or indirectly from a DB. User input that is not quoted and used directly in SQL can be interpreted as SQL.</td>
<td></td>
</tr>
<tr>
<td>CWE–78</td>
<td>Improper Neutralization of Special Elements used in an OS Command (“OS Command Injection”).</td>
<td>Not applicable because the application does not do direct calls to the shell or OS.</td>
</tr>
<tr>
<td>CWE–120</td>
<td>Buffer Copy without Checking Size of Input (“Classic Buffer Overflow”)</td>
<td>Not applicable because the code is written in Java without any native calls. It is not written in C or C++.</td>
</tr>
<tr>
<td>CWE–79</td>
<td>Improper Neutralization of Input During Web Page Generation (“Cross-site Scripting”)</td>
<td>Not applicable because there is no place to enter string–input that is displayed for another user.</td>
</tr>
<tr>
<td></td>
<td>This is about injecting active content that is displayed in a page that is generated for another viewer.</td>
<td></td>
</tr>
<tr>
<td>CWE–306</td>
<td>Missing Authentication for Critical Function.</td>
<td>Premise P4 is challenged because the identity of the PSP doing the call–back has to be checked.</td>
</tr>
<tr>
<td>ID</td>
<td>Name and Description</td>
<td>Premises Challenged</td>
</tr>
<tr>
<td>---------</td>
<td>--------------------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| CWE–862 | Missing Authorization  
An authenticated user can access unauthorized content. This vulnerability can be exploited when the server have functions with different levels of authorization and no control on them. | Premise P4 is challenged because PSP doing the call–back should not answer to other transactions belonging to other PSP.                                                                                               |
| CWE–798 | Use of Hard–coded Credentials                                                        | Premise P6 is challenged because hard coded credentials can be extracted. DB credentials can reveal the billing agreement.                                                                                             |
| CWE–311 | Missing Encryption of Sensitive Data                                                 | Premises P2, P3, P4 and P5 are challenged.                                                                                                                                                                            |
| CWE–434 | Unrestricted Upload of File with Dangerous Type                                      | Not applicable because there is no file upload functionality in the payment server.                                                                                                                                    |
| CWE–807 | Reliance on Untrusted Inputs in a Security Decision                                | Not applicable because there is no security decision based on user input in the payment server. The player has already logged in when making a payment. There is no special authorization to make a payment. No user roles are stored on the client. |
| CWE–250 | Execution with Unnecessary Privileges  
A security breach can be more severe with higher privileges. If higher privilege is requested for a specific function it has to be changed back after the function is executed. | Challenges premises P1, P2, P3 and P4 but not directly. This weakness can amplify other weaknesses.                                                                                                                  |
<table>
<thead>
<tr>
<th>ID</th>
<th>Name and Description</th>
<th>Premises Challenged</th>
</tr>
</thead>
</table>
| CWE–352| Cross–Site Request Forgery (CSRF)  
This is about tricking the user to click on something that makes a state changing request the user does not intend to and where the request is performed with the user credentials. | Not applicable because a user cannot be tricked into paying for someone else. The user is logged in to the system and all the payments go into the user’s own account.                                 |
| CWE–22 | Improper Limitation of a Pathname to a Restricted Directory (“Path Traversal”)        | Not applicable because there are no file operations using input from the user taking place in the payment server.                                                                                                   |
| CWE–494| Download of Code Without Integrity Check                                              | Not applicable because there is no download of remote code in the payment server.                                                                                                                                     |
| CWE–863| Incorrect Authorization  
This includes errors such as the role saved in the client cookie.                   | Not applicable because no authorization is required to make payments.                                                                                                                                                  |
| CWE–829| Inclusion of Functionality from Untrusted Control Sphere                               | Not applicable because the libraries used are stored locally outside of the web root.                                                                                                                                     |
| CWE–732| Incorrect Permission Assignment for Critical Resource  
This is about the permission to configuration files and other resources.                       | Premise P1 and P6 is challenged because an unauthorized person can get access to the local network and DB credentials and the billing agreement.                                                                      |
<p>| CWE–676| Use of Potentially Dangerous Function                                                 | Not applicable because the code is written in Java without any native calls.                                                                                                                                          |</p>
<table>
<thead>
<tr>
<th>ID</th>
<th>Name and Description</th>
<th>Premises Challenged</th>
</tr>
</thead>
<tbody>
<tr>
<td>CWE–327</td>
<td>Use of a Broken or Risky Cryptographic Algorithm</td>
<td>Premises P2 and P3 can be challenged if the encrypted transaction can be altered.</td>
</tr>
<tr>
<td></td>
<td>Some older algorithms are not as strong as thought including MD4, MD5, SHA1 and DES.</td>
<td>Premise P4 can be challenged if the result with checksum is altered.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Premise P6 can be challenged if the DB encryption is breached.</td>
</tr>
<tr>
<td>CWE–131</td>
<td>Incorrect Calculation of Buffer Size</td>
<td>Not applicable because the code is written in Java without any native calls.</td>
</tr>
<tr>
<td>CWE–307</td>
<td>Improper Restriction of Excessive Authentication Attempts</td>
<td>Not applicable because the players do not log in to the payment server. The log in functionality is handled by the game server.</td>
</tr>
<tr>
<td>CWE–601</td>
<td>URL Redirection to Untrusted Site (“Open Redirect”)</td>
<td>Challenges the trustworthiness of the site if the site can be used by hackers for phishing user sensitive information.</td>
</tr>
<tr>
<td></td>
<td>This weakness is exploited if user input is used to build up redirects. It can be used for phishing.</td>
<td></td>
</tr>
<tr>
<td>CWE–134</td>
<td>Uncontrolled Format String</td>
<td>Not applicable because the code is written in Java without any native calls.</td>
</tr>
<tr>
<td>CWE–190</td>
<td>Integer Overflow or Wraparound</td>
<td>Not applicable because the code is written in Java without any native calls.</td>
</tr>
</tbody>
</table>
ID | Name and Description | Premises Challenged
---|----------------------|-------------------------
CWE–759 | Use of a One–Way Hash without a Salt | Not applicable because the payment server does not handle user logins and does not create hash values.

5.6.3 OWASP Top 10

The CWE/SANS Top 25 and OWASP are not disjunctive sets. There is not a one to one mapping between them. OWASP focuses on Web–applications. The CWE/SANS Top 25 is “Most Dangerous Software Errors is a list of the most widespread and critical errors that can lead to serious vulnerabilities in software” as can be read in [20]. There is a mapping in there as well. The prioritizing is different but the end result is similar (See Table 5-2).

Table 5-2: OWASP top 10 in relation to CWE top 25.

<table>
<thead>
<tr>
<th>OWASP ID</th>
<th>Name and Description</th>
<th>CWE Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWASP–1</td>
<td>Injection</td>
<td>Includes both CWE–89 and CWE–78 but includes more.</td>
</tr>
<tr>
<td></td>
<td>All kind of injection that can be done to an interpreter. Includes also LDAP, Xpath, XML parsers, SMTP Headers, program arguments, etc.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This is about flaws in user logins and session management, timeouts etc.</td>
<td></td>
</tr>
<tr>
<td>OWASP–3</td>
<td>Cross–Site Scripting</td>
<td>Same as CWE–79</td>
</tr>
<tr>
<td>OWASP–5</td>
<td>Security Misconfiguration</td>
<td>CWE–250, CWE–732</td>
</tr>
<tr>
<td>OWASP ID</td>
<td>Name and Description</td>
<td>CWE Mapping</td>
</tr>
<tr>
<td>----------</td>
<td>----------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>OWASP–6</td>
<td>Sensitive Data Exposure</td>
<td>CWE–327, CWE–311</td>
</tr>
<tr>
<td>OWASP–8</td>
<td>Cross–Site Request Forgery</td>
<td>Same as CWE–352</td>
</tr>
<tr>
<td>OWASP–9</td>
<td>Using Components with Known Vulnerabilities</td>
<td>This is about human processes to make sure not to use bad components. Not covered by CWE.</td>
</tr>
<tr>
<td>OWASP–10</td>
<td>Unvalidated Redirects and Forwards</td>
<td>CWE–601</td>
</tr>
</tbody>
</table>

**5.7 Step 6: Classify Risks**

This step is to find the risks that should be mitigated by the system and the ones that should be mitigated by the system context.

Of the twenty five CWE there are nine that are applicable to the system.

CWE–250 execution with unnecessary privileges can be achieved in two ways. Either the system asks for more privileges to perform a function and then does not change the privileges back or the application is run with higher privileges than needed when deployed. The payment system does not ask the system for more privileges. The CWE can be transferred to the deployment environment.

CWE–732 Incorrect Permission Assignment for Critical Resource is about permissions on configuration files and other resources such as local network. This is about deployment can be transferred to the deployment environment.

CWE–327 Use of a Broken or Risky Cryptographic Algorithm can be transferred to the PSPs. They are the ones deciding what crypto algorithm to use.

The remaining CWEs are mitigated by the system.
5.8 Step 8: Verify Mitigation

This step is to verify that the mitigations are implemented correctly. I will verify that the risks are mitigated by finding weak spots in the information flow, doing penetration tests and code review. Detection methods for the CWEs can be found on the public CWE catalogue. Some CWE are best verified by doing code review and others are best verified by doing test.

5.8.1 Code Review

The scope of the code review includes the code for communicating with the PSPs.

The system is at the moment integrated to seven PSPs. There is a framework for the integration (see class diagram in Figure 5.3). When the player selects payment method a PurchaseService class creates the transaction in DB and calls a RedirectService class for the corresponding PSP for retrieval of a redirection link. The PSP–RedirectService class is responsible for PSP specific functionality. The RedirectService class for most PSPs initiates a transaction at the PSP for the specific amount and currency. Item information to display and the callback URLs are also passed to the PSP. The PSP returns a token that is included in the redirect URL or posted to the redirect URL.

The player is redirected to the PSP site. She/he enters payment details. The PSP verifies the details and when ready makes a callback directly to the merchant or redirects the player’s browser to the merchant callback URL or both. Most PSPs are done after this but there are some that require an acknowledgement from the merchant.

There is one servlet for each PSP that receives the callback from the PSP. In most cases the servlet just checks the callback parameters and calls the client. Some PSP expect to receive an ACK–message else they resend the callback a number of times. One PSP requires a debit call to actually make the payment.
Figure 5.2 shows the flow between the game server, payment server and the PSP. The vulnerable points are marked with stars in the diagram. They are the communication between the game server and the payment server and the communication between the payment server and the PSP.

Two vulnerable points concerns the communication with the game server. The game server communicates with the payment server using JavaScript Object Notation Remote Procedure Call (json-rpc). The payment server has to make sure the call is really coming from an authorized client. There is no authorization check in the code. It is handled by the infrastructure. The payment server accepts only calls from within the local network. This has to be tested when doing penetration tests.

The third vulnerable point is the redirect to the PSP. The payment system have to make sure when redirecting the player to the PSP that it is not possible to change the price or currency and that as little information as possible is exposed. For all PSPs except Zong a session is initiated before the redirect. The Zong PSP does not initiate a session or a transaction before the redirect. The price is in the URL. It might therefore be possible to change the URL to pay a lesser price. Because the price is not checked in the callback it might be possible to pay less. This has to be tested when doing penetration tests.

The fourth vulnerable point is the redirect from the PSP back to the payment server after the transaction has been processed. PayPal does not do a direct callback to the payment server. The callback goes through the customer browser as a redirect from the PSP to the server. It might be possible to change the redirect URL in order to use the transaction token from a cheaper transaction to pay for a more expensive transaction. This has to be tested when doing penetration tests.

The fifth vulnerable point is the callback from the PSP. The payment server has to check that the call is coming from the authorized PSP when receiving
the callback. In some cases the callback is signed which also protects the message from alteration. There is no authorization check in the code for the cases the message is unsigned. In those cases the infrastructure handles the authorization with HTTPS. The payment server has to also check that the price is correct and that the callback is for the correct transaction. This has to be tested when doing penetration tests.

Some CWE are better verified by doing code review and some are better tested by doing penetration tests. The best detection method is on the CWE itself. The code review on the CWE and OWASP that came out of step 6 is listed in Table 5-3.

Table 5-3: Code review comments on the CWE and OWASP applicable for the payment server.

<table>
<thead>
<tr>
<th>ID</th>
<th>Code Review Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>CWE–89</td>
<td>The code review did not find any injection points. The framework handles DB queries in a safe and consistent manner. The data stored in DB is parameterized. DB calls also use parameterized search parameters. SQL injection is not possible. An attacker that gains DB access could read or modify transactions already being made. Worst thing an attacker could do is to disable the PSPs resulting in a DoS attack, but there are no injection points as mentioned above.</td>
</tr>
<tr>
<td>ID</td>
<td>Code Review Comment</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CWE–306</td>
<td>The payment server does not handle user accounts or logins. It receives a request from the game server. The game server handles the user login and session. There are some credentials located in property files. The property files are not accessible from outside the local subnet. In order to access the DB an attacker has to have the credentials but the attacker has also to be located in the subnet. The payment server handles the user session from the PSPs that create a session for the payment. Code review shows that the session can be hijacked and reused for another payment. This is the case with PayPal that does the server callback through a redirect in the player’s browser. This has to be tested when doing penetration tests. The load balancer breaks the HTTPS and sends HTTP to the payment server. It does neither check the certificate nor does it check that the callback is coming from an authorized PSP. No authentication is done for call-backs from PSPs. This has to be tested when doing penetration tests.</td>
</tr>
<tr>
<td>CWE–862</td>
<td>This vulnerability is exploited when a user that is authorized to access a resource but changes the object reference and is able to access another object she/he is not authorized to access. The objects in the payment server handled by reference that could be subject to this kind of attack are the transactions. Changing the PSP transaction reference in the callback for one that is approved by the PayPal PSP is a possible scenario. There is a page that lists all transactions of the user. It is not possible to change the user reference to see the transactions of another user because the user id is retrieved from the session.</td>
</tr>
<tr>
<td>CWE–798</td>
<td>There are no hard-coded credentials in the code.</td>
</tr>
<tr>
<td>ID</td>
<td>Code Review Comment</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CWE–311</td>
<td>The sensitive data in the system is the payment details sent to the PSP and the result coming from the PSP. It has to be protected from alteration. The callback from the PSP are either signed or sent using HTTPS. The load balancer breaks the HTTPS and sends HTTP to the payment server. The communication between the load balancer and the payment server is within the protected local network.</td>
</tr>
<tr>
<td>CWE–250</td>
<td>The payment server does not change privileges during run time. It does not require access to privileged operating system resources.</td>
</tr>
<tr>
<td>CWE–732</td>
<td>This vulnerability is about the installation and configuration of software. The operations department makes sure that the latest patches are installed. They change the default passwords and make sure files are protected from access.</td>
</tr>
<tr>
<td></td>
<td>The program is not creating files or directories except for logging. Only software administrator should have write permissions on the configuration–files.</td>
</tr>
<tr>
<td>CWE–327</td>
<td>This is about protecting sensitive data with encryption. The payment server does not handle credit card numbers in clear text. The sensitive data is the transaction itself. It has to be protected from alteration in transit. The weak spot is the callback from PSP. Some PSP does not use TLS/SSL for this communication but have a signature to make sure the data is not altered. The public key used to verify the signature is stored locally. To make a fake signature one has to have control of the server. Other PSP use HTTPS but the load balancer breaks the TLS/SSL. The payment server makes a confirmation call in this case. The payment server does not do this for Payvision.</td>
</tr>
<tr>
<td></td>
<td>The game company cannot change the algorithm used for communicating with the PSPs. It is decided by the PSP.</td>
</tr>
<tr>
<td>ID</td>
<td>Code Review Comment</td>
</tr>
<tr>
<td>-------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CWE–601</td>
<td>This is about dynamic redirection where the URL is not validated. The user can then be redirected to a malicious site. The payment solution is based on redirecting the player to the PSP site for data entry. The redirection URL is not build up on data entered by the user. It is retrieved completely from the server. It resides either in the property files or in the DB. I think it is possible for someone to do a man in the middle attack and change the redirect for a specific user.</td>
</tr>
<tr>
<td>OWASP–9</td>
<td>The operations department patches the software with the latest patches. There is no process in place that guarantees that the libraries used in the code are up to date.</td>
</tr>
</tbody>
</table>

The code review resulted in four possible vulnerabilities to test doing penetration tests.

5.8.2 Penetration Testing

This step is for testing issues found during code review but also for testing the CWEs that are better tested with this detection method.

The majority of scanning tools test only 45 percent of all CWE in year 2007. They are also testing different things. The choice of tool is important says Hines Matt on the internet newspaper ComputerWorldUK [25].

For testing the vulnerabilities I need a tool that acts as a proxy to do a man in the middle attack. It has to be easy to inspect, filter and grab a request and then modify it and resend it. There are a handful of tools for testing doing this. I chose to use Burp because it has the required functionality and the company already uses it.

I tested to change the redirect URL to the Zong PSP. Instead I use a redirect URL for a smaller amount. The payment with a smaller amount worked because the payment server does not verify that the amount in the call–back is correct. I could buy the most expensive wares and pay the smallest amount.

CWE–89 is efficiently tested by doing automated static analysis. I did code review. Penetration testing is not a recommended method in this case because there could possibly be many fields for data entry.
CWE–306 is missing authentication for critical function. The PSP callbacks to the server are sent on HTTPS but the load balancer breaks the HTTPS. I want to test whether the balancer checks certificate and where the request comes from. To test this I send a callback in HTTPS from my Burp–client. As can be seen in Figure 5.4 the callback goes through to the payment server. An error message is displayed because the format of the message is incorrect. I could have approved any transaction if I had the correct format on my message.

![Image of Burp Suite interface](image-url)

*Figure 5.4: A fake callback request in the Burp client. The error message shows that the transaction got through to the payment server but the format is wrong.*

CWE–862 is missing authorization check when an actor tries to perform an action. I try changing the parameter “coreUserId” in the page to display the payment methods and the user transactions. It does not do any harm. I was not able to see data belonging to some another user.

CWE-862 PayPal sends the callback through the client browser as a redirect. I make a small payment and copy the redirect callback–message with the approved token. I use the approved message for my next payment and just change the parameter “transaction” with the transaction–id I want to be approved and it works (see Figure 5.5). It works for transaction belonging to other PSPs. There is no authorization check that makes sure a PSP cannot
accept a transaction belonging to another PSP. A token for an approved transaction is valid for three hours. It can be used for paying any transaction for three hours.

![Burp Suite Free Edition](image)

**Figure 5.5:** The redirect callback in the Burp client showing the transaction identifier and the token. The same token can be used to approve any transaction for three hours.

CWE–798, CWE–311, CWE–732 and CWE–601 are checked doing code review.

CWE–327 I did not try to break the SSL.

OWASP–9 is not verified neither doing code review nor doing penetration test. It is about processes and policies. There is no process in place to mitigate this.

The code review and penetration testing shows some vulnerabilities that should be addressed. I will discuss the results in section 5.11.
5.9 Step 9: Prioritize Risks

The step to prioritize the risks is for knowing what is most important to the system and to get a measurement of the security to be able to compare releases. I use the Common Weakness Scoring System (CWSS) for the priority calculation and the Common Weakness Risk Analysis Framework (CWRAF) for adding the deployed technology and business importance aspect to the calculation.

The first step in prioritizing using CWRAF is to specify a vignette. The vignette contains the archetypes and technical score cards. It specifies the technology for the system and the importance the technical impact has on business (See Table 5-4).

*Table 5-4: Vignette for the payment server.*

<table>
<thead>
<tr>
<th>Name</th>
<th>Payment Server</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Archetypes</strong></td>
<td>Database, Web browser, Web server, Transactional engine</td>
</tr>
<tr>
<td><strong>Business Value Context (BVC)</strong></td>
<td>Integrity is essential. Availability is also important. Security incidents might result in financial loss and brand damage.</td>
</tr>
</tbody>
</table>

The technical impact scorecards are all on application layer because it is not possible to gain control of the network, hardware or physical host by hacking the application. Here I put a score on the technical impacts based on the consequences it has to the business (See Table 5-5).
I calculate a new score for the CWE using the CWSS and CWRAF. CWSS considers 16 factors when calculating a score as described in sector 3.2.2. The factor technical impact in CWSS can be quantified using CWRAF. The CWE has a field for the common consequence in the public catalogue. Using the field I find the technical impact score cards. The score divided by 10 is used as score for the factor technical impact.

I calculate a new score for the CWEs applicable for the payment server. The calculation is done in Excel. The result is in Table 5-6.

### Table 5-5 Technical impact scorecards for the vignette.

<table>
<thead>
<tr>
<th>Technical Impact</th>
<th>Score</th>
<th>Consequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modify data</td>
<td>9</td>
<td>Delete or modify transactions</td>
</tr>
<tr>
<td>Read data</td>
<td>1</td>
<td>See what transactions have been made</td>
</tr>
<tr>
<td>DoS: resource consumption</td>
<td>5</td>
<td>Inability to process transactions</td>
</tr>
<tr>
<td>DoS: unreliable execution</td>
<td>5</td>
<td>Difficulty tracking whether transactions have succeeded or not</td>
</tr>
<tr>
<td>Execute unauthorized code or commands</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gain privileges / assume identity</td>
<td>3</td>
<td>Gaining admin privileges authorizes to inactivate PSPs</td>
</tr>
<tr>
<td>Bypass protection mechanism</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hide activities</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5-6: CWE with score.

<table>
<thead>
<tr>
<th>ID</th>
<th>Name and Description</th>
<th>Common Consequence</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>CWE–89</td>
<td>Improper Neutralization of Special Elements used in an SQL Command (&quot;SQL Injection&quot;)</td>
<td>Modify application data</td>
<td>0</td>
</tr>
<tr>
<td>CWE–306</td>
<td>Missing Authentication for Critical Function</td>
<td>Gain privileges / assume identity</td>
<td>32</td>
</tr>
<tr>
<td>CWE–862</td>
<td>Missing Authorization</td>
<td>Gain privileges / assume identity</td>
<td>32</td>
</tr>
<tr>
<td>CWE–798</td>
<td>Use of Hard-coded Credentials</td>
<td>Bypass protection mechanism</td>
<td>12</td>
</tr>
<tr>
<td>ID</td>
<td>Name and Description</td>
<td>Common Consequence</td>
<td>Score</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------</td>
<td>-------------------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>CWE–311</td>
<td>Missing Encryption of Sensitive Data</td>
<td>Modify application data</td>
<td>21</td>
</tr>
<tr>
<td>CWE–250</td>
<td>Execution with Unnecessary Privileges</td>
<td>Gain privileges / assume identity</td>
<td>6</td>
</tr>
<tr>
<td>CWE–732</td>
<td>Incorrect Permission Assignment for Critical Resource</td>
<td>Gain privileges / assume identity</td>
<td>7</td>
</tr>
<tr>
<td>CWE–327</td>
<td>Use of a Broken or Risky Cryptographic Algorithm</td>
<td>Modify application data</td>
<td>13</td>
</tr>
<tr>
<td>CWE–601</td>
<td>URL Redirection to Untrusted Site (&quot;Open Redirect&quot;)</td>
<td>Gain privileges / assume identity</td>
<td>0</td>
</tr>
<tr>
<td>OWASP–9</td>
<td>Using Components with Known Vulnerabilities</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The reason why the score for CWE-89 is zero is because the formula includes the factor “Internal Control Effectiveness” that is zero because as mentioned in code review it is not possible to exploit. CWE-601 is also zero for the same reason. The URL is retrieved from property file or the DB.

## 5.10 Fraud Prevention

There are mainly two kinds of perpetrators committing fraud to the system. There are the criminals that use stolen cards and then there are the players that want to get virtual goods in-game without paying. The case with stolen cards is a police matter. The company has to provide the police with the information they require. It is desirable to stop them already when the payment is being done.

The behavior of the fraudulent gamer is different. There are two kinds; the benign one makes a payment for playing more and then makes a cashback in the bank. He gets back the money from his bank and the in-game money is reduced. He then buys back the in-game money to continue playing. He is behaving fraudulent to get kind of credit. The malign fraudulent player is different. He creates an account, buy lots of in-game money resulting in lots of payments during short time and then make cashbacks in the bank. He gets the money back from his bank, the in-game money is reduced and his account is blocked. The cashback request reaches the company some weeks later. Before that the fraudulent player has synchronized his account with
Facebook and can continue playing there with the stolen in–game money. The company wants to identify these players and block them before they synchronize their account with Facebook. It is important not to block non–fraudulent users to avoid unnecessary calls to helpdesk. The system should propose players to be blocked but the decision is made by a human.

A report showing players and their transactions is needed. The report has to show the number of payments and amounts, number of different cards used and what IP–addresses. The players in the report have to be carefully selected. The selection criteria have to be based on previously known fraudulent behavior stored in the data warehouse (DWH). I get the transaction data from the DWH containing player identification, amount, date of transaction, currency and whether the transaction was cashback.

I investigated the possibility to use the hidden markov model to stop them but the method does not suite this case. The reason is that there is not enough information in the system. The hidden markov model requires previous transactions to configure the probabilities for the normal behavior of the specific player. Based on previous transactions the model calculates the deviation in probability. The normal behavior of the card holder is not in the DWH because most players make only a few payments. It might even be payments with someone’s card that never played before. The perpetrators with stolen cards create a new account and make many payments during short time with one or more stolen cards. They also try to hide their IP address using a VPN or trusted proxy.

In order to find the fraudulent behavior I study transaction amount, average amount per day and the number of transactions per day.

The binomial distribution is the probability of a specific number of successes in a sequence of independent yes or no trials. A player can be fraudulent (yes) or non–fraudulent (no). Every player can be fraudulent or non–fraudulent independent of the others. The distribution becomes binomial with the probability mass function (1) where \( n \) is total number of player, \( k \) is number of fraudulent players and \( p \) is the probability of a fraudulent player.

\[
\binom{n}{k}p^k(1-p)^{n-k}
\]

When sorting out the fraudulent players 10 % of false positives is an acceptable limit for the company because it will be handled manually.

The suspicion is that fraudulent players make higher amount payments than non–fraudulent players. 50 EUR is a very high amount in game payments so if fraudulent players make higher amount payments then they should be majority. I set the limit between higher amount and lower amount payments to 50 EUR. The hypothesis is then that more than 90 % of all the payments
higher than 50 EUR are charged back. The null hypothesis is that less than 90 % of all the payments higher than 50 EUR are charged back.

There were 938 payments over 50 EUR and 59 payments were charged back. Assuming there is a probability of 90 % for a chargeback the probability to only get 59 chargebacks is minimal using the binomial distribution. The probability of getting 59 or less is (2).

\[
\sum_{k=0}^{59} \binom{938}{k} 0.9^k 0.1^{938-k}. \tag{2}
\]

Calculating with the statistical program R gives result zero. The hypothesis cannot be proved and is probably wrong. The transaction amount cannot be used to identify charge backs.

In order to find the fraudulent behavior I study the normal behavior of the average amount per day for the users and make a histogram to see the distribution.

![Diagram over average amount per day in euro for non-fraudulent players.](image)

As can be seen in Figure 5.6 most users spend low amounts per day. It is explainable because the wares are sold in packages. The size of the packages varies depending on game. The most popular has packages of 1 EUR, 4 EUR, 8 EUR and 12 EUR. There are small spikes at these amounts in the histogram.

I make the same for the fraudulent transactions and compare.
Figure 5.7: Diagram over average amount per day in euro for fraudulent players.

For the fraudulent users the histogram looks a little bit different (See Figure 5.7). A bigger share of fraudulent users spends more than 20 EUR per day. 32 % of fraudulent users spend more than 20 EUR compared to 4 % of other users.

I state the hypothesis that of the players that spend more than 50 EUR per day 90 % make refunds. The null hypothesis is that less than 90 % of the players that spend more than 50 EUR per day make refunds.

There are 177 users that spend more than 50 EUR per day and 22 fraudulent users that spend more than 50 EUR per day. It is 12.4 %. The hypothesis can be rejected. The analysis on amounts per day shows no correlation between spending amount per day and refunds. The spending amount per day alone cannot be used to identify fraud.

As can be seen in Table 5-7 the amount per day cannot be used.

Table 5-7: Amount per day.

<table>
<thead>
<tr>
<th>Amount per day</th>
<th>&gt;20 EUR</th>
<th>&gt;30 EUR</th>
<th>&gt;40 EUR</th>
<th>&gt;50 EUR</th>
</tr>
</thead>
<tbody>
<tr>
<td>No of normal users</td>
<td>1071</td>
<td>517</td>
<td>252</td>
<td>155</td>
</tr>
<tr>
<td>No of fraudulent users</td>
<td>48</td>
<td>35</td>
<td>27</td>
<td>22</td>
</tr>
<tr>
<td>Total no of users</td>
<td>1119</td>
<td>552</td>
<td>279</td>
<td>177</td>
</tr>
<tr>
<td>% of fraudulent users</td>
<td>4.3%</td>
<td>6.3%</td>
<td>9.7%</td>
<td>12.4%</td>
</tr>
</tbody>
</table>
A suspicion is that fraudulent users make more transactions during a short
time than other users. The histogram over the number of transactions per
day for normal users in Figure 5.8 shows that most users make one or two
transaction per day in average.

![Number of Transactions per Day](image)

*Figure 5.8: Diagram over number of transactions per day for normal users.*

To see if this criterion can be used I set a limit and compare the number of
transactions per day for fraudulent and all users. The hypothesis is that of
the users making more than ten transactions per day at least 90 % are fraud-
ulent. The null hypothesis is that of the users making more than ten transac-
tions per day less than 90 are fraudulent.

There are 152 users making more than ten transactions per day and 13 of
them are fraudulent users. It makes 8.6 %. The hypothesis can be rejected.

As can be seen in Table 5-8 the number of transactions per day cannot be
used even for very high limits.

*Table 5-8: Statistic for number of transactions per day.*

<table>
<thead>
<tr>
<th>Transactions per day</th>
<th>&gt;5 EUR</th>
<th>&gt;10 EUR</th>
<th>&gt;15 EUR</th>
<th>&gt;20 EUR</th>
</tr>
</thead>
<tbody>
<tr>
<td>No of normal users</td>
<td>754</td>
<td>139</td>
<td>46</td>
<td>28</td>
</tr>
<tr>
<td>No of fraudulent users</td>
<td>29</td>
<td>13</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>Total no of users</td>
<td>783</td>
<td>152</td>
<td>54</td>
<td>31</td>
</tr>
<tr>
<td>% of fraudulent users</td>
<td>3.7%</td>
<td>8.6%</td>
<td>14.8%</td>
<td>9.7%</td>
</tr>
</tbody>
</table>
There is a difference in number of transactions between these two user groups (See Table 5-9). Fraudulent players tend to do more transactions in average. The standard deviation is also very high. This alone cannot be used to distinguish this group. The mean value of 47 transactions for fraudulent users is matched by 565 non–fraudulent users doing between 45 and 50 transactions.

Table 5-9: Statistic data on number of transactions.

<table>
<thead>
<tr>
<th></th>
<th>Fraudulent</th>
<th>Non-fraudulent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>47.88</td>
<td>18.81</td>
</tr>
<tr>
<td>Median</td>
<td>14</td>
<td>4</td>
</tr>
<tr>
<td>Standard Deviation</td>
<td>116.60</td>
<td>60.67</td>
</tr>
<tr>
<td>Range</td>
<td>980</td>
<td>4664</td>
</tr>
<tr>
<td>Minimum</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Maximum</td>
<td>981</td>
<td>4665</td>
</tr>
</tbody>
</table>

The information I have is the transaction date, amount, currency and player id. I tried to see if I can distinguish fraud from regular behavior by looking at the statistics on transaction amount, amount per day and number of transactions per day. It turns out that it cannot be used as criteria for distinguishing fraudulent users. The distribution within the group is different than the distribution for normal users as can be seen in Figure 5.6 and Figure 5.7, but the number of charge back transactions is so small compared to the normal transactions that they cannot be filtered out. The prevalence of charge back transactions is 0.4 % and all the tested hypothesis gives large false positives.

There was no time to do some more advanced association analysis for a combination of parameters such as linear discriminant analysis (LDA) or logistic regression analysis. LDA requires that the independent variables are normally distributed while logistic regression does not. We cannot assume normally distributed variables. Using binary logistic regression analysis the dependent variable would be whether the transaction is fraudulent and the independent variables would be the transaction amount, amount per day and number of transactions per day.

5.11 Results
The functional requirements with the payment flow were written down.

Security requirements were found.

A list with the most important CWE for this application was assembled.
The verification step resulted in three vulnerabilities.

CWE-306: missing authentication on the callbacks from PSP. I recommended that an IP-address filter has to be added.

CWE-862: The callback from Paypal goes as a redirect via the customer browser. The session can be stolen and reused during three hours for any transaction. My recommendation was to modify the code handling the callback so as to add a check that the PSP result is for the correct transaction.

For the PSP Zong it was possible to change the redirect URL and pay less because there is no check for the price in the callback code. I recommended adding a check on the price in the code receiving the callback from Zong.

The fraud prevention effort gave unfortunately no result.
6 Conclusions

Payment systems are described in chapter 2. Payments in-game are conceptually not different than other internet payments. Payment service providers are used by merchants to handle the payment transaction.

Threats and security requirements for online payment systems were found and are described in chapter 3. Authentication, integrity, confidentiality, non-repudiation of transaction, privacy and accessibility are requirements for payment systems. There are many security threats such as tapping and sniffing messages, hacking, malicious code attacks and DoS attacks.

Frameworks and systems for measuring security were found and are described in chapter 3. There is an open framework (CWRAF) for considering business value when measuring security vulnerabilities that can be used together with a grading system (CWSS), both community developed. There are security standards for different businesses. For companies handling cards there is the PCI-DSS standard. There is a big database with vulnerabilities (CWE).

A method for analyzing the payment server of the gaming company was found and modified to suite the case. This is explained in chapter 4. I show in chapter 5 how the modified method can be used to analyze the security of a system. The functional requirements were written down in a consistent way. The security assets were identified. Security requirements were found and threats in the form of arguments were found. The risks were identified and classified. A verification was done by doing code review, information flow analysis and penetration testing.

The company has a security department that regularly does black-box penetration testing and sometimes code reviews on the APIs. They hire a big international company specialized and expert in security for the black-box testing. My work on information flow analysis and code review show that black-box penetration testing is not enough. I found three serious issues that were analyzed and corrected.

A statistical analysis was done to find fraudulent users in section 5.10. The transaction amount, average amount per day and the number of transactions per day were studied. Even when there seems to be a pattern in user behavior the fraud prevention statistical analysis shows that the prevalence of fraudulent users makes it impossible to sort them out without having a huge amount of false positives.
7 Discussion

I think it is beneficial that the whole development team have security in mind and not only the security experts. I also think it is important to have the security in thought throughout the development cycle. For using this method in the development cycle it has to be accommodated. When developing functional requirements there is a step for testing. There should also be a step to verify the mitigations for testing the security requirements in the development cycle.

The chosen method includes a step for prioritizing of the risks. The reason for this step is to end up with a good enough security because you can never be sure to have a 100% secure system. I think there should be one more prioritizing step after classifying the risks. This prioritizing should be very lightweight. Prioritizing should be done before mitigation so that you mitigate the highest priority risks first because normally you do not have unlimited resources and time.

I think that the security concerns should be present during the whole development cycle. The method can be modified to be incorporated in the development process used by the company.

The company uses Scrum, an agile process. The cycle is two week long sprints. There is a sprint planning meeting the first day of the sprint and the scope is set for the sprint. The tasks to do are described in the backlog. They are often minimal descriptions of functional requirements. Each developer tests his/her own implementation and demonstrates it at the end of the sprint. There is no step for system test. Continuous releases are a goal but today releases are made when needed. A retrospective meeting is held last day of the sprint to improve the process.

In order to incorporate the method into the development process step 1 and step 2 should be done when creating the task in the backlog. The description of the task is then more complete. Step 3 and step 4 should be done when analyzing the task after it is selected to be in the sprint. The security requirements and outer arguments should be a natural step when analyzing what to do. Steps 5, 6 and 7 should be done when implementing the task. Knowledge of the CWE and how to mitigate them is acquired over time and improves the quality of the end product. This might lead to changes in the requirements and then the previous steps have to be reviewed. It is natural to test the security requirements when testing the functional requirements. That is why I added a verification step in the method. At the end of the sprint the team or managers might want to know whether the security improved or worsened during the sprint. Step 8, the risk priority calculation is performed.
Following the steps as described above involve the whole development team in the security work. The security of the system is thought of during the whole development cycle. In this way secure systems are built.
Bibliography


