BACHELOR’S THESIS

Security of small business e-commerce sites

Author: Karel Kohout
Supervisor: Ing. Jaromír Veber
Academic Year: 2009/2010
Declaration of Authorship

The author hereby declares that he compiled this thesis independently and that all sources have been included in the list of literature and cited according to the ČSN ISO 690 norm.

Prague, September 2, 2010

Signature
Čestné prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Veškeré použité podklady, ze kterých jsem čerpal informace, jsou uvedeny v seznamu použité literatury a citovány v textu podle normy ČSN ISO 690.

V Praze, 2. září 2010

__________________________

Podpis
Acknowledgments

The author is especially grateful to his advisor, Ing. Jaromír Veber, for invaluable comments and assistance with the thesis and to Ms. Alison Dansie for English language counseling.
Abstract

The thesis describes selected security issues affecting small business e-commerce sites in the European Union and also shows whether such issues exist in authentic applications currently being used by means of a small scale study.

It presents several specifics of e-commerce, a summary of European Union (European Community) law with focus on private data protection and unsolicited mail based on literature search and an overview of minimum security requirements for web-based applications. The research part contains an assessment of parts of a source code, documentation, incident response procedures and installation scripts of seven most popular PHP open-source shopping carts with regard to security derived from criteria and directives defined in the theoretical part. A limited field study of possible effects of security (or of lack thereof) on turnover is conducted and results are available at the end of the thesis.

Keywords  computer security, electronic commerce, open-source shopping cart, private data protection

Author’s e-mail xkohk02@vse.cz
Supervisor’s e-mail qvebj00@vse.cz
Abstrakt

V bakalářské práci jsou popsány vybrané problémy spojené se zabezpečením malých internetových obchodů v zemích Evropské unie. Formou studie zkoumá, zda se zmíněné problémy vyskytují v reálně používaných aplikacích.

Na základě podkladové literatury rozebírá některá specifika elektronického obchodování, evropské komunitární právo se zaměřením na ochranu osobních údajů a nevyžádaná obchodní sdělení a uvádí přehled minimálních požadavků na zabezpečení webových aplikací. Výzkumná část obsahuje zhodnocení sekcí zdrojových kódů, dokumentace, instalačních skriptů a řešení bezpečnostních incidentů sedmi nejpoužívanějších PHP open-source e-shopů s ohledem na požadavky stanovené v teoretické části. Prezentovány jsou i výsledky menší studie možného efektu zabezpečení na obrat internetového obchodu.

Klíčová slova

bezpečnost, elektronické obchodování, open source e-shop, ochrana osobních údajů

E-mail autora

xkohk02@vse.cz

E-mail vedoucího práce

qvebj00@vse.cz
Contents

List of Tables ix

List of Figures x

Acronyms xii

1 Introduction 1

1.1 Goals of the thesis .................................. 2
1.2 Characteristics of e-commerce ...................... 2
  1.2.1 E-commerce sites as a target ................... 8
  1.2.2 Applicability of security and IT frameworks .... 11
1.3 Limitations of the thesis .......................... 11

2 Theoretical requirements 13

2.1 Legal framework .................................... 13
  2.1.1 Protection of private data ...................... 13
  2.1.2 General business rules ......................... 17
  2.1.3 Unsolicited messages ............................ 18
2.2 Technical requirements .............................. 20
  2.2.1 Definition of assets ............................. 20
  2.2.2 Implications of HTTP protocol ................. 22
  2.2.3 User authentication and accounts ................ 26
  2.2.4 Input sanitation ................................. 29
  2.2.5 Encryption .................................. 33
  2.2.6 Logs, data retention, backups and others ...... 34

3 Security of selected open-source e-commerce applications 36

3.1 Criteria and methodology ................................ 38
3.2 Software selection .................................. 41
3.3 Assessment ........................................ 44
  3.3.1 Magento ................................... 46
3.3.2 OpenCart ................................................................. 47
3.3.3 Oxid e-shop community edition ................................. 48
3.3.4 Prestashop ............................................................... 48
3.3.5 Ubercart ................................................................. 49
3.3.6 VirtueMart ............................................................... 50
3.3.7 Zen Cart ................................................................. 53
3.4 Remarks ................................................................. 55

4 User perception of security ............................................. 57
  4.1 Survey questions and results ...................................... 58
  4.2 Remarks ................................................................. 63

5 Conclusion ............................................................... 65

Bibliography .............................................................. 71

A SSL performance impact ............................................... I
B Eavesdropping – traffic statistics ................................... III
C PAP intercepted communication ................................... IV
D Test server configuration ............................................. VI
E Zen Cart – session tokens ........................................... XI
F Payment methods ..................................................... XIII
G Survey screenshots .................................................... XIV
H Content of Enclosed DVD ........................................... XVII
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Types of computer crimes</td>
<td>10</td>
</tr>
<tr>
<td>2.1</td>
<td>Security stakes</td>
<td>21</td>
</tr>
<tr>
<td>3.1</td>
<td>Software selection overview</td>
<td>43</td>
</tr>
<tr>
<td>3.2</td>
<td>Assessment results</td>
<td>45</td>
</tr>
<tr>
<td>4.1</td>
<td>Skill levels</td>
<td>58</td>
</tr>
<tr>
<td>4.2</td>
<td>Age ranges</td>
<td>58</td>
</tr>
<tr>
<td>4.3</td>
<td>Time spent using computers</td>
<td>59</td>
</tr>
<tr>
<td>4.4</td>
<td>SSL recognition</td>
<td>60</td>
</tr>
<tr>
<td>4.5</td>
<td>User tracking</td>
<td>62</td>
</tr>
<tr>
<td>D.1</td>
<td>Test server configuration</td>
<td>VI</td>
</tr>
<tr>
<td>F.1</td>
<td>Payment methods, Germany</td>
<td>XIII</td>
</tr>
</tbody>
</table>
## List of Figures

1.1 System context diagram of a typical e-commerce solution ............... 4  
1.2 Non-obvious attackers of EC sites .................................. 8  
2.1 Definition of terms – 95/46/EC ........................................ 14  
2.2 Controller requirements .................................................. 15  
2.3 Safe harbor principles .................................................... 16  
2.4 Minimum information ..................................................... 17  
2.5 Order acknowledgement .................................................. 18  
2.6 Unsolicited communications .............................................. 19  
2.7 Session tokens (GET, POST) .............................................. 23  
2.8 Examples of “leaked” session tokens ................................... 23  
2.9 PHP session identifier generator ........................................ 25  
2.10 Django session identifier generator .................................... 25  
2.11 LSO, PostAffiliate Pro 4.x ............................................... 27  
2.12 User rights, ISO/IEC 17799 ............................................... 29  
2.13 Reflected XSS example ................................................... 31  
2.14 Redirect attack ............................................................. 32  
2.15 CSRF protection .......................................................... 33  
2.16 Disabled certificate validation ........................................... 35  
3.1 Database setup ............................................................. 41  
3.2 Magento – altered tmp dir ............................................... 46  
3.3 Magento – temporary files permissions ................................ 46  
3.4 VirtueMart – URL rewriting filter ....................................... 51  
3.5 VirtueMart – SQL injection I .............................................. 52  
3.6 VirtueMart – SQL injection II ............................................ 52  
3.7 Zen Cart, .htaccess ......................................................... 53  
3.8 Zen Cart – password hashes .............................................. 54  
3.9 Zen Cart – HTML escaping ............................................... 54  
3.10 Comparison of best and average software package ................. 56
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.1</td>
<td>File – 10240 Bytes</td>
<td>I</td>
</tr>
<tr>
<td>A.2</td>
<td>Encrypted vs non-encrypted communication</td>
<td>II</td>
</tr>
<tr>
<td>B.1</td>
<td>Networking equipment malfunction – traffic</td>
<td>III</td>
</tr>
<tr>
<td>D.1</td>
<td>PHP configuration</td>
<td>VII</td>
</tr>
<tr>
<td>D.2</td>
<td>PHP configuration, cont.</td>
<td>VIII</td>
</tr>
<tr>
<td>D.3</td>
<td>Suhosin configuration</td>
<td>IX</td>
</tr>
<tr>
<td>D.4</td>
<td>Apache 2 virtual host configuration</td>
<td>X</td>
</tr>
<tr>
<td>E.1</td>
<td>Zen Cart – session token regeneration I</td>
<td>XI</td>
</tr>
<tr>
<td>E.2</td>
<td>Zen Cart – session token regeneration II</td>
<td>XII</td>
</tr>
<tr>
<td>G.1</td>
<td>Facebook – non-SSL login page</td>
<td>XIV</td>
</tr>
<tr>
<td>G.2</td>
<td>Facebook – SSL login page</td>
<td>XV</td>
</tr>
<tr>
<td>G.3</td>
<td>Firefox 3 – self-signed certificate warning</td>
<td>XV</td>
</tr>
<tr>
<td>G.4</td>
<td>Altered web page screenshot</td>
<td>XVI</td>
</tr>
</tbody>
</table>
Acronyms

3DES  Triple DES (encryption standard based on DES – Data Encryption Standard).

AES  Advanced Encryption Standard (encryption standard based on Rijndael).

API  Application Programmable Interface.

ASCII  American Standard Code for Information Interchange (character set).

ASP  Active Server Pages, in the thesis also ASP.NET

B2B  Business To Business.

B2C  Business To Customer.

CA  Certificate Authority.

CAPTCHA  Completely Automated Public Turing test to tell Computers and Humans Apart (spam preventing measure).

CMS  Content Management System.

COD  Call On Delivery (customer pays when physically receiving the goods).

Conversion rate  Rate at which visitor accomplishes preset action. Typically percentage of visitors who complete an order.

CRM  Customer Relationship Management (also CRM software).

CSRF  Cross-Site Request Forgery.

DoS  Denial of Service.

DDoS  Distributed Denial of Service.

DNS  Domain Name System.

EC  Electronic Commerce.
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERP</td>
<td>Enterprise Resource Planning (also ERP software).</td>
</tr>
<tr>
<td>EU</td>
<td>European Union.</td>
</tr>
<tr>
<td>HIPAA</td>
<td>Health Insurance Portability and Accountability Act (US legislation).</td>
</tr>
<tr>
<td>HTML</td>
<td>Hyper Text Markup Language.</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hyper Text Transfer Protocol.</td>
</tr>
<tr>
<td>IDS</td>
<td>Intrusion Detection System.</td>
</tr>
<tr>
<td>IT</td>
<td>Information Technology.</td>
</tr>
<tr>
<td>MD5</td>
<td>Message-Digest algorithm 5, cryptographic hash function (see RFC 1321).</td>
</tr>
<tr>
<td>MSIE</td>
<td>Microsoft Internet Explorer.</td>
</tr>
<tr>
<td>MX</td>
<td>DNS record used for SMTP protocol routing (email).</td>
</tr>
<tr>
<td>NIST</td>
<td>National Institute of Standards and Technology, Department of Commerce.</td>
</tr>
<tr>
<td>PPC</td>
<td>Pay-Per-Click (method of paying for advertisement, i.e. Google Adwords).</td>
</tr>
<tr>
<td>RAID</td>
<td>Redundant Array of Independent Disks.</td>
</tr>
<tr>
<td>REST</td>
<td>Representational State Transfer.</td>
</tr>
<tr>
<td>RNG</td>
<td>Random Number Generator.</td>
</tr>
<tr>
<td>SHA-1</td>
<td>Secure Hash Algorithm 1, cryptographic hash function (see RFC 3174).</td>
</tr>
<tr>
<td>SHA-2</td>
<td>Secure Hash Algorithm 2, cryptographic hash function (see FIPS 180-2).</td>
</tr>
<tr>
<td>SMB</td>
<td>Small-Medium Business.</td>
</tr>
<tr>
<td>SOX</td>
<td>Sarbanes-Oxley act (US legislation).</td>
</tr>
<tr>
<td>SPF</td>
<td>Sender Policy Framework (see RFC 4408).</td>
</tr>
<tr>
<td>SQL</td>
<td>Structured Query Language</td>
</tr>
<tr>
<td>SSL</td>
<td>Secure Sockets Layer (method of encryption of traffic).</td>
</tr>
<tr>
<td>TCP-IP</td>
<td>Internet Protocol Suite, communication protocol.</td>
</tr>
</tbody>
</table>
**TLS**  Transport Layer Security (successor of SSL).

**URL**  Uniform Resource Locator.

**US**  United States (United States of America).

**VAT**  Value Added Tax.

**XML**  Extensible Markup Language.

**XSS**  Cross-Site Scripting.

**zero-day exploits**  Also 0-d attack; “an exploit of a vulnerability for which a security update does not exist” (source: http://msdn.microsoft.com/en-us/library/cc307402.aspx, last retrieved June 18, 2010).
Chapter 1

Introduction

The security of web applications in general is becoming an important issue of today, as vulnerable sites present a vector for both client side and server side attacks. Due to technological innovations and economic recession, the number of individuals for whom income from criminal activity in ICT (and hence economic motivation) is equal to or larger than possible loss is growing [9, pg. 16]. At the same time, the marginal cost of crime has been reduced by “specialization and sophistication” [9, pg. 16], and automated frameworks can be used with limited skills to effectively leverage known bugs. With decreased costs, low and medium profile websites are becoming a viable target.

In such an environment, protecting business applications is vital and already well established in case of medium to large businesses, as they have been a target since the 90’s (or even earlier). Security of web sites is a well discussed topic too, with numerous studies available to any possible reader, ranging from causal users to ICT professionals. However, small e-commerce sites, a synthesis of business applications (where large scale security frameworks are financially infeasible) and of public websites (with increased security requirements) are considerably less researched. The current writer aims to study a selected part of this area: the security of small e-commerce applications in the European Union.

Sources have been chosen to cover the topic from all possible aspects – theoretical security background ([4]), web applications security ([6]), software assessment procedures ([7]) and security of a particular group of software in the evaluation ([32]). Additional sources are the Directives of the European Parliament and Council (legal), ISO/IEC norms and NIST publications (800 series, FIPS) as industry “best practices”, OWASP project (top threats) and various research papers. Because of the abundance of literature covering each segment, newer publications were preferred over older, if possible, and the author would like to kindly point out that due to the sheer amount of publications, a complete search of literature is impossible.
The thesis is structured as follows: Chapter 1 contains a general introduction, description of the field of study, and limitation of the thesis. Chapter 2 examines theoretical and technical requirements of a secure e-commerce site. Chapter 3 studies how selected requirements are fulfilled (or supported) by common open-source e-commerce software. Finally Chapter 4 aspires to evaluate if compliance with Chapter 2 affects turnover in a limited field study.

The next section contains goals of the thesis, an overview of differences between offline and online business, e-commerce security specifics, and a definition of an e-commerce application.

1.1 Goals of the thesis

The objective of this thesis is to describe the most common security issues and vulnerabilities found in small business B2C e-commerce sites, with focus on sites in the European Union, and to demonstrate whether the vulnerabilities can be exploited in “real-life” applications and also to show whether e-commerce security can possibly affect turnover from the customers’ view. The issues explored should range from technical ones (SQL injection) to legal ones (compliance with local, European Union’s “acquis communautaire” and international law).

Goals will be achieved by describing the legal perspective and general security problems of web applications and by evaluating popular open-source e-commerce packages (shopping carts) against a set of criteria selected by the author. The effect of security on turnover will be examined in a small scale exploratory study with a limited set of participants.

1.2 Characteristics of e-commerce

When comparing companies selling “offline” and “online”, there are many similarities and as many differences. Apart from location – online business is seemingly international\(^1\), whereas “offline” business is at a fixed place (places for chain store), and character of goods (“virtual goods” are not typically sold in brick-and-mortar stores), one of the main differences that will be important throughout the thesis is the ability to exactly measure costs (and the cost of measuring costs).

In offline business (especially retail), one can measure how many customers have entered the store, the size of the order, and therefore one can study which sociodemographic groups typical buyers belong to. However, extensive studies are expensive,

\(^1\) Many risks are associated with over-the-border selling in the European Common Market (e.g. warranty, certain items falling under extensive excise duties legislation or VAT).
and hence they cannot be performed on a daily basis and even with the best studies, only aggregate results can be achieved. On the contrary, on e-commerce sites, any individual visitor can (and is) tracked from the landing page (and through referer fields of HTTP requests even one page before the point of entry) through the site, checkout process and payment. With user accounts, it is possible to track individual visitors’ habits (and recommend products). By adding the costs of certain actions, it is possible to directly (and exactly, with a very small margin for error) assess the cost of selling each product, attracting each visitor to the page, and from the point of security, to precisely experiment with and measure costs of any security arrangements (e.g. whether adding a “seal of trust” increases conversion rate, the trade-off between enforcing strict string formatting – thus lowering conversion rate – and paying more staff to correct malformed strings in orders and so on).

Other differences are no less important. Online vendors are extremely dependent on third parties – search engines, review sites, affiliates, shipping companies etc. From the customers’ point of view, buying online is also very specific. One’s ability to act on word of mouth has been greatly extended (online review sites, blogs, social networks), as has been the ability of “window-shopping”. A certain effect of “commodisation” of goods can be observed, especially in certain markets (computer hardware, books, consumer electronics) – customers can easily search for a chosen brand and model and thus buy at the lowest possible price, because the marginal cost of searching (the cost of going through another store) is extremely low compared to normal stores. Role of intermediaries is pronounced – one could even use the term “hypermediation” to correctly describe the current state.

From the point of security, three key differences, paraphrased according to NIST [37, pg. 2-1], are:

**Increased Efficiency:** Due to automation, the marginal cost of one extra attack is very small. This means that an attacker can have a very low success rate of attack, yet still be profitable (similar to the effectiveness of spam).

**Action at a Distance:** Any attacker can attack any website from any part of the world with sufficient connectivity. Local law enforcement and local institutions will inevitably be ineffective.

**Rapid Technique Propagation:** Especially with the use of common software, zero-day

---

2Subsection 2.2.2 discusses several problems associated with tracking visitors.
3Loyalty programs, e.g. (credit)cards are used for the same purpose.
4Chapter 4 tries, at a limited scope, to assess the effect.
5Dependence on search engines may seem to be similar to dependence on prime locations (streets, ...) However, for “prime spots” (first few results), there are several hundred competing companies.
6Institutions as in Hayek’s cosmos and taxis (Law, Legislation and Liberty).
exploits spread very quickly in a matter of days and the vendors may not be able (or willing) to provide patches in timely manner. Also, potential criminals may use the Internet for sharing knowledge.

Another key difference might be added to the previous three:

**Multi-layered attacks:** Most attacks do not aim for a single goal – current criminals are highly organized, and the attackers may not be those who actually leverage the attack – they may merely sell the acquired data to third parties, who will abuse it (log-in information, identity theft, phishing, spreading malware and others).

Defining what an actual e-commerce solution is presents another set of problems. As Figure 1.1 shows, typical e-commerce application must interact with several completely different “partners”. Also, the application itself must fulfill multiple roles and the distinction of where the e-commerce application ends and where other IT systems begin can be hazy.

**Figure 1.1:** Simplified system context diagram of a typical SMB e-commerce solution

Source: Author.

---

1It might be tempting to call the e-commerce application “the front end” and the rest (ERP, CRM) “the back end”. However, such separation is not clear either.
Depending on the company, the level of integration differs – an e-commerce site may be completely integrated (to the point where it is just another “view” of the ERP/CRM system), or it can be connected very loosely (all the systems are running separately and there is some sort of data export/import filter\(^8\)). For the purpose of the thesis, a small e-commerce solution should fulfil the basic needs of e-commerce:

- order processing,
- shipping,
- invoicing,
- limited payment processing (at least COD),
- basic CRM functions,
- basic business analytic functions.

Accounting or warehouse management functions may or may not be part of the solution – if not, the expected integration shall be limited\(^9\). The context of a small company, where a full-blown IT system is not financially feasible and an improvised integration is far cheaper, must not be forgotten.

Customers in an SMB e-commerce solution go through two distinct processes – selecting goods (arriving at the site, browsing the site, adding goods to the shopping cart and moving on to checkout) and checkout (inputting address, selecting shipping, rechecking the order, submitting the order, optionally finishing the payment sub-processes and receiving order confirmation).

Accessibility is very important – except specialized B2B stores, for standard B2C interaction, use of specialized software is uncommon\(^10\). The usual minimum requirement is a recent web browser (at the time of writing of the thesis – 2009/2010 – this means MSIE 6+, Opera 9.x+, any Gecko 1.9+ based browser – for example Firefox 3.x+, any WebKit based browser – Chrome, Safari, Konqueror), support for cookies and JavaScript. Adobe Flash may sometimes be employed for design purposes, but it is rarely necessary to complete the order. In special cases (e.g. printing digital photographs), a desktop application or a browser-based Java client may be required.

From the point of computer security, characteristics differ depending on the user. For staff (clerks, managers, analysts), typical rules of computer security apply (strong password, encryption). For customers, the lowest common denominator must be taken into account. The e-commerce site must support the oldest browser

---

\(^8\)Such integration can be achieved by e.g. daily XML feeds, running SQL queries from one system’s database against another, REST API, in external systems via parsing emails (e.g. bank account statements) etc.

\(^9\)By limited, imagine one way XML export from e-commerce software to accounting software or use of a simple API. Tight integrations is in some cases impossible – e.g. accounting software in personal versions running only on office computers with Windows only during work days.

\(^10\)Example (but not in the SMB sector): iTunes.
and the least computer-literate customers and the actual process of order should be extremely simplified. Hence, enforcing typical security solutions as PKI or multilayer authentication is out of question (Chapter 2 deals with the issues).

Also, for e-commerce, the basic information security dimensions, so-called CIA Triad – “Confidentiality, Integrity, Availability” [18, pg. 7] do not provide fine-grained control. An extended system [4, pg. 40 - 46], applied on an e-commerce site, supplies a better approach (notes added by the current writer):

**Availability** – e-commerce is a 24/7 business – any downtime costs money (and such costs can be exactly measured in lost orders). This also means that any e-commerce software should degrade “gracefully” – in a controlled manner, helpful to the visitor of the site.

**Integrity** – correct and unmodified state and detection of modification – does not only concern immutability of certain accounting data (e.g. invoices must not change when the price of products, shipping or tax rates are altered), it presents a large issue for SMB e-commerce solutions. Any information displayed to the customer throughout shopping (and after placing the order) should be the same, ranging from tax rates (a common problem of open-source solutions developed in the US when dealing with European-style VAT – see Drupal plugin Ubercart 1.x) to legally required information (conditions of use, privacy notice). Hence, e-commerce solution should provide a reasonable level of consistency. Also, an attack against e-commerce site should not affect the integrity of the solution. Low integrity implies limited trustworthiness of the data in the system. On the other hand, it is viable to allow selected company staff to change certain data because of flexibility required by customers.

**Authenticity** of e-commerce site owner can be determined by customers and is often legally required. Methods of improving the perceived authenticity are discussed in Chapter 4. Authenticity as in determining the correctness of the sender of the message from the point of site owner is generally impossible. It can be assumed that certain information is correct (address if the customer paid before the goods have been shipped, name and email for online payment methods), but the site owner has no other way to check the information provided by customers. Technical measures should be employed to verify that the message has been sent, with high probability, by a human (CAPTCHA and similar methods, CSRF protection) and telephone may be used in case of “call on delivery” payment. None of those measures can prevent the so-called “delivery race” – a situation when customer orders at several sites, the one that
ships the fastest wins (customer pays COD fee), the rest lose (site owners pay postal fees).

**Non-repudiation** is unachievable in B2C e-commerce, because most transactions are not legally binding and/or provable, except for contracts verified by a third party (e.g. online payment). For B2B with custom software, client side certificates or other technology and a written contract make non-repudiation possible. It also plays an important role when dealing with shipping companies.

**Confidentiality** is usually required by the law (storing customer information). Technologies for achieving confidentiality are described in detail in Subsection 2.2.5.

**Non-observability** (third party cannot determine whether any exchange of messages is taking place, or – a weaker form – cannot identify two senders) usually does not occur in e-commerce transactions (and would be very difficult to achieve with TCP-IP).

**Anonymity** causes a lot of friction – store owners are bound by the law to protect personal information, yet for business purposes, identifying visitors is vital. The requirement can be exploited, to a certain degree, by both parties (customers, merchants) to their advantage – by transferring the cost of storing sensitive data (credit cart information) to a third party (online payment system).

**Accountability** – tracing actions back to respective individuals – plays an important part in the e-commerce landscape. Any action should be logged, both for placing checks at the staff and for general purposes (“who changed what”).

**Evidence** can be seen as a part of accountability. For example, at least some retention of traffic logs is mandatory in many countries of the world. As far as e-commerce software goes, evidence is often collected at the lowest meaningful level – web or database server and as such is out of the scope of the thesis.

**Temporal correctness** is often neglected, but significantly affects the turnover. Consider delivery time of emails (order confirmation, invoices), time for confirmation of payment (too long a time will cause the transaction to be cancelled). Interesting issues arise with large changes dependent on time – see recent change of VAT in the Czech Republic, when orders placed at 11:59 December 31, 2009 would be taxed by 9 %, whereas orders placed at 0:01 on January 1, 2010 by 10 %. It may seem mundane, but with a sufficient delay, a discrepancy in VAT arises.
Separation of roles is a common security requirement. In the context of e-commerce, a fine-grained system of roles is required.

Covert obligations concern auction sites (automatic bidding), but as a concept play only a minor role in e-commerce.

Fair exchange is crucial for the merchant to avoid legal litigation and to maintain store reputation.

Monitoring and Eavesdropping may be required by law enforcement, but are usually covered by collecting evidence (access logs) and in the context of e-commerce are not meaningful (possibly except for selling certain special goods like weapons and ammunition).

1.2.1 E-commerce sites as a target

Considering who and why people attack e-commerce solutions produces interesting findings. Classification of all attackers is difficult (due to the fact that most of them fall into multiple categories) and not all of them may be “crackers” (or “hackers” in general media) who attack just for financial gain. A few selected categories from [15, pg. 34] show some non-obvious reasons why an e-commerce site may be attacked:

*Figure 1.2: Non-obvious attackers of EC sites*

| Cyberterrorists | Terrorists who employ hacker-type techniques to threaten or attack against systems, networks, and/or data (…) |
| Hacktivists     | Hackers who break into computer systems in order to promote or further an activist agenda. (…) |
| Script kiddies  | Individuals with fairly limited hacking skills who rely upon scripts and programs written by other, more competent, hackers. Hackers of this type typically cause mischief and malicious damage (…) |

Source: [15, pg. 34].

The last, but not the least possible attacker in the category is a professional cracker, employed either by competition or by a criminal gang (see [39, pg. 296]). Attackers from “less developed countries” should not be underestimated either, as “technological change, the increased specialization and sophistication in the production of malware, and the globalization of the information and communication industries have all reduced the marginal cost of crime. In turn, this cost decrease has dramatically expanded the supply of crime, as people from countries and regions with low opportunity cost of labour\(^{11}\) (which increase the net benefits of crime) join

\(^{11}\)Intuitively: capital more expensive than labour.
criminal activities. Such reduced marginal costs of security violations will shift the marginal cost of crime schedule downwards. Assuming that other things, especially the benefit relationship, remain unchanged, reductions in the marginal cost of crime will result in a higher level of security violations and vice versa.” [9, pg. 16]

We can derive other motives than monetary gain – mainly political agenda (especially dangerous to sites selling animal products, cosmetics, pharmaceutics, books related to religious views) and social acceptance (computer “vandalism”). Those motives may seem to be superficial, but they set minimum security requirements for many parts of the website (professional attackers value their time and correctly assess the risk of detection, whereas most “script-kiddies” do not).

To complete the overview, a list of possible crimes related to e-commerce follows.
### Table 1.1: Types of computer crimes

<table>
<thead>
<tr>
<th>Crime / abuse</th>
<th>Description</th>
<th>Notes (examples)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Fraud</strong></td>
<td>For private gain or benefit (altering input in an unauthorised way, destroying, suppressing, misappropriating computer output, altering computerised data, alteration or misuse of programs (excluding virus infections)).</td>
<td>Changing bank accounts (or PayPal addresses) for customer payments; altering prices of products.</td>
</tr>
<tr>
<td><strong>Theft</strong></td>
<td>Of data; of software.</td>
<td>Affiliate software, e-commerce software, customer data, traffic statistics.</td>
</tr>
<tr>
<td><strong>Use of unlicensed software</strong></td>
<td>Using illicit copies of software.</td>
<td></td>
</tr>
<tr>
<td><strong>Private work</strong></td>
<td>Unauthorised use of the organisation’s computing facilities for private gain or benefit.</td>
<td>Using company’s web hosting to share private data.</td>
</tr>
<tr>
<td><strong>Misuse of personal data</strong></td>
<td>Unofficial ‘browsing’ through computer records and breaches of data protection legislation.</td>
<td>Staff, IT administrators.</td>
</tr>
<tr>
<td><strong>Hacking</strong></td>
<td>Deliberately gaining unauthorised access to a computer system, usually through the use of communication facilities.</td>
<td>Either to commit other crimes or just “for fun”.</td>
</tr>
<tr>
<td><strong>Sabotage</strong></td>
<td>Interfering with the computer process by causing deliberate damage to the processing cycle or to equipment.</td>
<td>Ex-employees, competitors, DDoS.</td>
</tr>
<tr>
<td><strong>Introducing pornographic material</strong></td>
<td>Introducing pornographic material, for example, by downloading from the Internet.</td>
<td>Placing inappropriate material can quickly cause the website to be shut down.</td>
</tr>
<tr>
<td><strong>Virus</strong></td>
<td>Distributing a program with the intention of corrupting a computer process.</td>
<td></td>
</tr>
</tbody>
</table>

*Source: [15, pg. 30], also in [27], [39, pg. 246]. Notes (third column): author.*
Assessing what an actual attack is is far more complicated – not all security problems related to e-commerce are classifiable as a crime. Ranging from a public website “defacement” to collecting publicly available information\(^{12}\), an attack is, for the purpose of the thesis, anything deliberate (or, at least, as a result of neglecting responsibilities) directly or indirectly affecting site turnover.

### 1.2.2 Applicability of security and IT frameworks

Staff size and budget severely limit the security of small e-commerce sites and the applicability of “enterprise level” security frameworks (especially for applications audit [8]). Some inspiration is provided by ISO/IEC 27000 series (Information technology – Security techniques – Information security management systems), ISO/IEC 12119 (Information technology – Software packages – Quality requirements and testing), ISO/IEC 15408 (Information technology – Security techniques – Evaluation criteria for IT security) and others. COBIT or ITIL as whole systems are probably too complicated, but certain parts of ITIL (e.g. Service Design – Business impact analysis) could be used to improve the management of security (and IT in general) of the e-commerce site (ITIL 2 contained a Small-scale implementation book, which is not part of ITIL 3).

### 1.3 Limitations of the thesis

The thesis solely focuses on small e-commerce software (even though certain parts apply to all electronic commerce sites) mainly in the European Union, with some overlap to the US (this limit is set for practical purposes by the impossibility of evaluation of all legislation combinations). For evaluation, only open source (or mixed-license) applications were chosen – lack of source code (or source code available under non-disclosure agreement) would prevent the author from achieving the goals of the thesis. Examples in the thesis are generally valid, however, they are simplified and mostly demonstrated on certain operating systems (GNU/Linux) and in applications written in certain selected languages (Python, PHP) due to the author’s familiarity with such technologies. There might be many other technologies (based on e.g. Microsoft products or other Unices, other languages, other libraries) that prevent certain specific attacks or, on the contrary, provide a new attack surface.

Only the actual e-commerce applications are studied, but, as research papers suggest, “the successful functioning of e-commerce security depends on a complex interrelationship between several applications development platforms, database man-

\(^{12}\)See Subsection 2.2.4
agement systems, systems software and network infrastructure."[20] Other systems, relevant to security, are omitted in the thesis (IDS systems, systems preventing DoS and password guessing\textsuperscript{13}, firewalls, ... Online payment systems, integration with analytical and marketing tools and back end systems are omitted too, as are backups and other issues generally related to computer security (disaster preparedness, personal security, etc. – for a complete list, see e.g. [18]). General information security guidelines aimed at small companies can also be found in a NIST publication [22].

As for sources of data, the author collected general information about visitors (to assess current browsers, etc.) from two distinctive websites – a restaurant in the centre of Prague, whose website is mainly visited by office workers before noon to read the daily menu (verified by hostnames) and serves as a small sample of the typical “corporate” (banks, telecommunications, lawyers) visitor. The second website is an e-commerce site of a company selling experiential gifts (off-line targeting team-building and employee benefits), which is a very competitive market with customers who do not always focus on price. For obvious reasons, only aggregate data may be disclosed. It can be argued whether the data is valid, but the author believes the set represents a useful sample of “buying” customers in the Czech Republic and is more representative than typical public statistics with unknown sources (e.g. in comparison to data collected from a small free hosting, with web pages mostly visited by avid PC gamers, browser brands were vastly different, as they are if one studies e.g. Toplist\textsuperscript{14}).

Local (Czech) market specifics are mentioned only where appropriate. The two key differences between Czech and European (or American) markets are a strong preference\textsuperscript{2} for call-on-delivery payment (instead of online payments – compare with statistics for Germany in Appendix F) and a duopoly of search engines Seznam.cz (ads – Sklik, product comparison – zbozi.cz, 60 % share including affiliated sites\textsuperscript{16}), Google (ads – AdSense/AdWords, Google Checkout, 30 % share\textsuperscript{16}).

The thesis continues with overview of theoretical security requirements.

\textsuperscript{13}mod_evasive, Fail2ban, DenyHosts and others.
\textsuperscript{14}A popular “counter” of visitors in the Czech Republic with public statistics, \url{http://toplist.cz/global.html}. 
Chapter 2

Theoretical requirements

First, a basic legal framework, based on European Union’s legislation (“acquis communautaire”) with focus on private data protection, will be laid out (a report as to why protecting private data is of such importance can be found in [30]). Then selected technical areas related to e-commerce security will be described in detail.

2.1 Legal framework

The “landscape” of e-commerce legislation by itself is extremely diverse, so the main focus will be in three areas: protection of private data, general business rules with compliance directly affected by security and unsolicited messages.

2.1.1 Protection of private data

First, let’s compare European and US privacy laws: the overall achieved result is similar, but the means differ. The EU system is based on single legislation covering all sectors including most online issues meet by citizens, whereas US legal system “uses a sectoral approach that relies on a mix of legislation, regulation, and self regulation” [36, Annex Im, par. 1] (e.g. HIPAA, SOX). “European” laws are implemented by member states, and a common framework is established at a community level. Hence, only directives are cited and each member state may implement each directive through different means and with small variations. Despite such variations, the EU “acquis communautaire” law provides a concise way to enumerate e-commerce legal requirements related to security.

The 95/46/EC Directive [11]\(^1\) implements the basis for private data protection in EU. Local implementations in member countries vary – for example Czech

\(^1\)Directive is projected in local laws, e.g. Czech 101/2000 Sb. (http://ec.europa.eu/justice_home/fsj/privacy/law/implementation_en.htm, last retrieved June 5, 2010). A more general approach (and, in an older version, a foundation for the directive) is provided by Article 8 of the
law 101/2000 Sb. [29] further diversifies data into personal and sensitive data\(^2\), but the general concept (notification, informed consent, maintaining data quality, maintaining data only for a period required for the processing, existence of a controller) remains the same.

**Figure 2.1: Definition of terms – 95/46/EC**

| **personal data** (a) | shall mean any information relating to an identified or identifiable natural person ("data subject"); an identifiable person is one who can be identified, directly or indirectly, in particular by reference to an identification number or to one or more factors specific to his physical, physiological, mental, economic, cultural or social identity; |
| **processing of personal data** (b) | shall mean any operation or set of operations which is performed upon personal data, whether or not by automatic means, such as collection, recording, organization, storage, adaptation or alteration, retrieval, consultation, use, disclosure by transmission, dissemination or otherwise making available, alignment or combination, blocking, erasure or destruction; |
| **controller** (d) | shall mean the natural or legal person, public authority, agency or any other body which alone or jointly with others determines the purposes and means of the processing of personal data; (…) |
| **processor** (e) | shall mean a natural or legal person, public authority, agency or any other body which processes personal data on behalf of the controller; |
| **third party** (f) | shall mean any natural or legal person, public authority, agency or any other body other than the data subject, the controller, the processor and the persons who, under the direct authority of the controller or the processor, are authorized to process the data; |
| **recipient** (g) | shall mean a natural or legal person, public authority, agency or any other body to whom data are disclosed, whether a third party or not; (…) |
| **the data subject’s consent** (h) | shall mean any freely given specific and informed indication of his wishes by which the data subject signifies his agreement to personal data relating to him being processed. |

*Source: [11, Article 2].*

Based on Figure 2.1 it is clear that any owner of an e-commerce site falls into the categories of controller and processor and subcontracts multiple processors (even the most basic processing of customer data for analytical purposes is data processing). Therefore, any such owner must comply with requirements in Figure 2.2.


\(^2\) "sensitive data" shall mean personal data revealing nationality, racial or ethnic origin, political attitudes, trade-union membership, religious and philosophical beliefs, conviction of a criminal act, health status and sexual life of the data subject, as well as any biometric data of the data subject;" [29]
2. Theoretical requirements

Figure 2.2: Controller requirements

- (...) the controller or his representative must provide (...) with at least the following information, except where he already has it:
  - (a) the identity of the controller and of his representative, if any;
  - (b) the purposes of the processing for which the data are intended;
  - (c) any further information such as
    - the recipients or categories of recipients of the data,
    - whether replies to the questions are obligatory or voluntary, as well as the possible consequences of failure to reply,
    - the existence of the right of access to and the right to rectify the data concerning him

Source: [11, Article 10].

The requirement seems simple enough, however, collection of data begins before the subject can be informed (depending on the perception of IP addresses\(^3\)). Moreover, it is impossible for the typical user to give an informed consent if complicated technologies are involved, or if data collected prior to the consent are linked with data after the consent had been given (tracking orders, affiliate software, etc.) With outsourced backups\(^4\) in mind (or backups at the server level, out of control of the site owner), this puts a requirement to keep information about any manipulation with customer data (mainly deletion), to maintain at least limited perception of any data protection. The requirement of informed consent is usually fulfilled during the checkout process (applies to all directives presented later on too).

So far, only protection of personal data in the EU has been mentioned. As for transfer of personal data for processing from EU to third countries, permission is granted only “if, without prejudice to compliance with the national provisions adopted pursuant to the other provisions of this Directive, the third country in question ensures an adequate level of protection”[11, Article 27, 1)]. To simplify the compliance for a common transfer (EU-US), there exists the “Safe harbor” (see Figure 2.3) process [36]\(^5\), which allows US companies to certify compliance with [11]. The actual certification can be self-regulatory (by joining an existing self-regulating privacy program or by developing their own). The benefits of safe harbour are assured from the date of self-certification of adherence to the Principles of Safe

---

\(^3\)For information about identification through browser, see Panopticlick, Electronic Frontier Foundation, https://panopticlick.eff.org/ (last retrieved February 22, 2010).

\(^4\)For full-scale backup, the customer data disappears at the time of the next backup. For incremental backup, one can only make an educated guess.

\(^5\)Note: can be explicitly overruled in selected cases by member states. See http://www.export.gov/safeharbor/ (last retrieved April 2, 2010), similar process is available for Switzerland: http://www.export.gov/safeharbor/swiss/eg_main_018498.asp (last retrieved April 2, 2010).
harbor to the Department of Commerce (or designees)\(^6\). The adherence itself may be limited by law enforcement requirements, national security etc. and manual data entry is exempt from any requirements.

**Figure 2.3: Safe harbor principles**

| Notice | An organization must inform individuals about the purpose of collection and use of the data. It must also supply a contact information for further inquiries. |
| Choice | Possibility to opt-out if information is to be disclosed to third party or used for other purpose than collected for. For sensitive information (“i.e. personal information specifying medical or health conditions, racial or ethnic origin, political opinions, religious or philosophical beliefs, (...) or information specifying the sex life of the individual”, explicit opt-in is required. |
| Onward transfer | Transfer to third-parties is only possible if third party complies with the safe harbor principles. Removes responsibility if third party violates the principles. |
| Security | “Organizations creating, maintaining, using or disseminating personal information must take reasonable precautions to protect it from loss, misuse and unauthorized access, disclosure, alteration and destruction.” |
| Data integrity | Personal information must remain relevant (“reliable for its intended use, accurate, complete, and current”). |
| Access | Individuals must be “able to correct, amend, or delete that information where it is inaccurate, except where the burden or expense of providing access would be disproportionate to the risks to the individual’s privacy in the case in question, or where the rights of persons other than the individual would be violated.” |
| Enforcement | Existence of mechanism enforcing compliance, including sanctions in case of failure. |

*Source:* [36, Annex I, par. 9 - 16], paraphrased.

The self-certification process is reasonably simple (the organisation provides contact details\(^7\), description of activities and its privacy policy, including dispute mechanisms and employee training) to the certification body and either assesses compliance through internal process or hires a third party.

**Directive on privacy and electronic communications** [10] further complements [11]. It applies to “the processing of personal data in connection with the provision of publicly available electronic communications services in public communications networks in the Community” [10, Article 3, par. 1]. Most of the directive is concerned with internet service providers, but Article 13 par. 2 is relevant to e-commerce. It explicitly allows use of email addresses of customers, collected in

---


\(^7\)See [http://www.export.gov/safeharbor/eg_main_018239.asp](http://www.export.gov/safeharbor/eg_main_018239.asp) (last retrieved April 2, 2010).
accordance with [11] in the context of sale, for marketing of other, similar products as long as certain rules are followed (see Subsection 2.1.3).

### 2.1.2 General business rules

*Directive on electronic commerce* [13], as the name suggests, establishes a common approximation of Member States’ laws. The directive specifically allows providing “information society services from another Member State” (except for certain special reasons). Article 5 sets the minimum information that must be supplied by the service provider:

![Figure 2.4: Minimum information](image)

(a) the name of the service provider;
(b) the geographic address at which the service provider is established;
(c) the details of the service provider, including his electronic mail address (...);
(d) where the service provider is registered in a trade or similar public register, the trade register in which the service provider is entered and his registration number, or equivalent means of identification in that register;
(e) where the activity is subject to an authorisation scheme, the particulars of the relevant supervisory authority;
(f) (...)
(g) where the service provider undertakes an activity that is subject to VAT, the identification number (...)

*Source:* [13, Article 5, par. 1].

These requirements for minimum information are enforced by Member States in different laws, but the directive establishes a common framework. Article 9 provides validity for contracts concluded by electronic means. Unfortunately, it is not possible to effectively prove whether such contract had ever existed or whether it had been made in “clear mind”, so it does not eliminate “delivery race” with COD payment. Article 11 contains an important requirement regarding the confirmation of the order. Usually, order confirmation is achieved by sending an email and a disruption of email delivery is a serious hazard (emails caught up by spam filter, non-deliverable emails, etc.)

---

8Local implementations can be found at [http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:72000L0031:EN:NOT](http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:72000L0031:EN:NOT) (last retrieved April 3, 2010). They vary in creation or modification of one law (e.g. Belgium, Sweden, United Kingdom), through the average of 5-6 laws to the extreme of 28 (Czech Republic).
Figure 2.5: Order acknowledgement

- the service provider has to acknowledge the receipt of the recipient’s order without undue delay and by electronic means,
- the order and the acknowledgement of receipt are deemed to be received when the parties to whom they are addressed are able to access them.

Source: [13, Article 11].

There are several possibilities to disrupt email delivery. The most common one is a spam filter, which from the point of view of the site owner, malfunctions and mistakenly marks legitimate messages as spam. This typically happens with:

**Malformed headers** Email headers not formatted according to RFC, invalid sender address, missing MX DNS records (in the future SPF records too).

**HTML emails** While HTML allows formatting the text according to corporate identity, it is also a significant characteristic of most spam. The email should always contain a plain-text version.

**Incorrect encoding** Certain servers prefer local encoding (e.g. cp-1250, iso-8859-2) to the de-facto standard (Unicode, utf-8/utf-16). Certain languages (i.e. Czech) contain special non-ASCII characters, that, when encoded improperly, trigger spam filter rules. This can also cause confusion of customers using certain web-based free email services, as their emails would be unreadable.

**Sending spam** can occur either knowingly – bulk mail – or unknowingly through shared web hosting (where another site with the same IP sends spam), abused script on the site (typically contact forms) and so on.

**Improper configuration** Certain mail servers deny delivering mail during the first attempt as an anti-spam measure (legitimate sender tries to deliver the email again after a set period). Improperly configured mail transfer agents may not hold the email in queue till the delivery.

### 2.1.3 Unsolicited messages

Every e-commerce site directly depends on email (either for confirmation of orders or for marketing purposes). Sending marketing emails is allowed under the provision of Article 13 of [10], as mentioned earlier.

---

9Notably centrum.cz.

10From the author’s experiments e.g. Czech seznam.cz, centrum.cz.
2. Theoretical requirements

Figure 2.6: Unsolicited communications

- 1. The use of (…) electronic mail for the purposes of direct marketing may only be allowed in respect of subscribers who have given their prior consent.

- 2. Notwithstanding paragraph 1, where a natural or legal person obtains from its customers their electronic contact details for electronic mail, in the context of the sale of a product or a service, in accordance with Directive 95/46/EC, the same natural or legal person may use these electronic contact details for direct marketing of its own similar products or services provided that customers clearly and distinctly are given the opportunity to object, free of charge and in an easy manner, to such use of electronic contact details when they are collected and on the occasion of each message in case the customer has not initially refused such use.

- 4. In any event, the practice of sending electronic mail for purposes of direct marketing disguising or concealing the identity of the sender on whose behalf the communication is made, or without a valid address to which the recipient may send a request that such communications cease, shall be prohibited.

Source: [10, Article 13].

While compliance with the directive for a single email address is possible, most e-commerce site owners possess large databases of accumulated contacts (mostly customers, but also business partners, addresses collected through general inquiries, etc.) The e-commerce application thus should support separating user email addresses by means of collection (e.g. addresses collected through logging of recipients of “send-to-a-friend” feature are risky for marketing). However, such advice would lower the turnover, as some of the recipients who did not give a consent might be interested in the offer. Hence, another, more viable, although also illegal, option, is employable on the basis that it cannot be easily proved, by a single person, that he/she did not sign up for a newsletter or inquiry. Two databases are needed – one of all the addresses collected by conducting business, and a second, of persons who directly opted-out of receiving marketing information. Thus, no person shall receive unsolicited email after expressing the disapproval, the marketing database is fillable with basically any email addresses (to a reasonable amount) and the possibility of legal damage is limited, as most users receive some information from e-commerce sites and either ignore it or read.

This strategy is useless in case of purchasing databases of customer data – some customers may insert email addresses with a suffix (john.doe+shop_domain@domain.tld) and with multiple uses of the database for different campaigns, the source of data will be easily discovered. A safer solution is to outsource the marketing campaign (pay for emails to be delivered to customers).

A database of customer emails constitutes the largest security liability of any e-commerce site with limited possibility of outsourcing (the second largest being credit
cart details, that can be eliminated by use of payment gateways and technologies like 3D Secure). Any successful breach of security resulting in access of the database will certainly have the emails as a prime target. In case of e-commerce, the addresses will most probably be valid, and owned by customers using the internet for shopping (therefore not afraid of buying online). Such list would have a significant price at the black market and can be utilized to discredit the site too by sending notifications to customers (“your data is not safe”), by requesting money for not sending any notification (ransom) or by endangering the site itself legally.

Despite not employing a full-scale security management framework such as ISO/IEC 2700x, site owners still should prepare a set of announcements and procedures and designate responsible employees in case of any security breach to avoid contradicting statements and inadvertent disclosure (denial) of the breach and its extent.

2.2 Technical requirements

2.2.1 Definition of assets

A simplified list of assets\(^\text{11}\) to define security requirements is established as follows (note: items not listed in order of importance; items adapted from [17]; some assets are expected to be provided through contracts):

- **Tangible assets:**
  - server hardware and networking equipment required to run the site
  - computer hardware used to administer the site
  - other assets not related to the actual site (warehouse, office building, etc.)

- **Intangible assets**
  - e-commerce site software (“shopping cart”), its customization
  - design of the site
  - domain name
  - DNS records
  - site email accounts\(^\text{12}\)
  - product database (structured list of products with current prices)

---

\(^\text{11}\)Due to the nature of small e-commerce sites, companies running them and the scope of the thesis, most physical assets are omitted.

\(^\text{12}\)Email accounts will not be described in the thesis as they are not related to the actual e-commerce site. However, isolating the email server (or provider) from the site is sensible in case of downtime – by using one provider for the site and another (geographically separated) for emails, the chance that at least one system will be accessible by customers can be increased. Data security (especially legal liabilities) must be taken into account in case of outsourcing outside Europe.
2. Theoretical requirements

- orders database (typically required to be immutable)
- customer database (addresses, credit cards, relation to orders)
- third party software related to the site (affiliate software, accounting software import modules, visitor tracking software)
- advertisement data
- visitors statistics, tracking database
- customer trust, brand name
- customer-created content (e.g. product comments, ratings)
- search engine results ranks for important keywords
- money stored in pre-paid services (e.g. AdWords)
- server and networking equipment software required to run e-commerce site software\(^{13}\)
- other software and data not directly related to the site.

While the accounting value of the assets may be relatively small, for a successful site the hard-to-appraise assets grow in importance. A failed RAID array of a server may be unfortunate, but the cost of loosing search engine positions is much more devastating\(^{14}\). Unfortunately, it is not possible to make even an educated guess as to what might be the costs of such assets. The estimate must be made on per-site basis and carefully updated. The same applies to all other business data (be it the knowledge of keywords with high conversion rate, unique algorithm to recommend related products or a list of loyal customers). “Direct losses should not be seen as representative of the overall problem. It would be much more devastating, for example, if online fraud eroded customer trust.”\(^ {9}\) Another view of assets is presented in Table 2.1:

\textbf{Table 2.1: Stakes Matrix: Cost of failing a security requirement stakes in $/Hour (2009)}

<table>
<thead>
<tr>
<th>Stakeholders</th>
<th>Confidentiality</th>
<th>Integrity</th>
<th>Availability</th>
<th>Non-repudiation</th>
<th>Authenticity</th>
<th>Privacy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer</td>
<td>10</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>6</td>
<td>12</td>
</tr>
<tr>
<td>Merchant</td>
<td>120</td>
<td>70</td>
<td>140</td>
<td>110</td>
<td>105</td>
<td>6</td>
</tr>
</tbody>
</table>

\textit{Source:} [3, table 1]; technical and financial intermediaries omitted.

\(^{13}\)Expected to be outsourced, and thus out of the scope of the thesis.

\(^{14}\)Especially in case of local search engines. Redirecting a site from one sub-domain to another (in the context of the same domain, e.g. \texttt{aaa.domain.com} to \texttt{bbb.domain.com} via HTTP 301) causes a mild drop in Google.cz for several hours, however, as an example, in the local Seznam.cz the same redirect moves the page for key keywords several hundred positions backwards for up to five weeks. (…)

A second view of assets should be evaluated too – assets as liabilities. The legal background has already been presented in Section 2.1. By anecdotal evidence, most small sites do not consider such liabilities to be of any importance, even though by accumulating private data, site owners may be subjected to penalties in amounts much higher than is the cost of any of the assets. Therefore a customer database as a whole is an asset (list of prospective buyers), but particular data present extreme liability (customer emails, customer habits and customer passwords) that should be vigorously protected. Any asset should thus be protected both because of its value and because of potential cost in case of a security breach.

In next subsections, selected techniques of attacks against e-commerce sites, assorted by the current writer from [28], [32], [34] and [6], will be presented and problems related to online e-commerce will be discussed. The description begins with methods of identification of requests from users and continues with authentication, encryption and finally, threats related to user input are briefly mentioned. If not specified, the behaviour of PHP functions has been verified in [1]. Most common techniques have been verified by the author and if possible, applied to situations valid for e-commerce applications.

2.2.2 Implications of HTTP protocol

As per the thesis limitations, only HTTP-based e-commerce solutions and with regards to current browsers, only HTTP/1.1 capable platforms are expected. The protocol itself is stateless, which means that the actual task of tracking users is typically done at a higher level – programming language / web server output filter or programming framework. The common way are unique identifiers (session tokens) either sent as a request variable (GET/POST requests) or stored in a client cookie (and also sent with every request).

In Figure 2.7, a brief login sequence of PHPBB 3 has been captured and serves as a clear example, as it uses both options. Because of the way sessions are used at a basic level (e.g. in PHP), no checking is done against user IP address, user agent or any other data and this allows “roaming” (changing IP’s) users to stay logged-in for indefinite time (very useful – enables the storage of users’ shopping carts). Unfortunately, it also means that with session identifiers in URL users can easily allow anybody to access their accounts (e.g. by sharing a link on Facebook or sending it by email to a friend, which is common for e-commerce sites). Such shared...

---

15 Users are likely to use the same password for multiple accounts and ultimately, for their email account, which may be used to restore passwords for most online services and to collect data for social engineering; see Chapter 4.
2. Theoretical requirements

**Figure 2.7:** Session token passed via GET and POST (truncated) requests

```
http://www.domain.com/index.php?sid=b954284101722f28efd55d6807e55a52
POST /ucp.php?mode=login&redirect=%2Fucp.php%3Fmode%3Dlogin HTTP/1.1
Host: www.domain.com
Cookie: phpbb3_59nnu_sid=b954284101722f28efd55d6807e55a52
Content-Type: application/x-www-form-urlencoded
Content-Length: 116
username=Name&password=Password&sid=b954284101722f28efd55d6807e55a52
&...
```

*Source:* Author, PHPBB3.

Session does not only affect site security, but also trustworthiness (two users sharing the same cart, data, etc. and probably in contact). Session tokens in cookies are much safer, as it is impossible for the user to willingly send his “session” to somebody else.

**Figure 2.8:** Examples of “leaked” session tokens

```
1.2.3.4 [12/Feb/2010:14:35:48 +0100] "GET /logo.gif?jane.doe@tiscali.cz HTTP/1.1" 200 3421 "http://mail71.tiscali.cz/mail/VerifyGet?sid=7D530DC17165440D274277A6D14F380A179A837D&userid=jane.doe@tiscali.cz&seq=I&auth=ICZUVYKZUCHYA3RC"
```

*Source:* Author, emailing campaign, server logs, anonymised, truncated.

Figure 2.8 presents leaked session tokens matched with user accounts. The original source is an HTML email with embedded image (*logo.gif*) and a GET parameter (email address), employed to estimate the percentage of emails read during a marketing campaign. Upon opening, the web-based email interface sent session token in the URL which was captured in a referer field. Theoretically\(^{17}\), an attacker can successfully access email accounts just with the session token. While the example attack is via email against webmail service, other ways of capturing session id’s are easily deviseable: session fixation (attacker forces user to visit the page / log-in with known session token), insecure connection (e.g. wi-fi), on-page scripts etc. (note: with JavaScript, cookies can be captured too, but only within the same domain policy).

Two other reasons support the preference of cookies over request variables, both

\(^{17}\text{Note: the author did not test whether email accounts were accessible with captured sessions, as that would be highly unethical and also a criminal offence in the Czech Republic. Centrum.cz leak has been fixed as of February 2010.}\)
2. Theoretical requirements

important for e-commerce. So called “clean”\textsuperscript{18} URLs, with high probability, improve search engine ranks (and are also easier to remember for users) and cookies (with embedded session variables) allow sharing the load over multiple application servers\textsuperscript{19}. Unfortunately, cookies are also vulnerable to multiple attacks (ie. cross-site scripting).

Session identifiers are used by users as a token to identify themselves. However, after assigning the id after authentication (which will be described later), the server (e-commerce software) has no way to authenticate the user again, so session id’s work as a sort of “one-time” password, which creates the requirement of uniqueness and unpredictability, in order that the potential attacker cannot predict the content of session identifier (and thus impersonate other users or even administrators after enumerating a few possible values). Several sources of entropy are used, typically a hash of an information supplied by the server (current time), by the user (IP address) and a pseudo-random number. Randomness and unpredictability of identifiers is crucial for the security.

Software authors may not be aware whether a (cryptographically) secure function generates session identifiers (e.g. CakePHP uses original PHP \texttt{session\_id()}, even though sessions are handled by the framework itself). Two examples were selected. One relatively imperfect (Figure 2.9) and one significantly more secure (Figure 2.10).

\textsuperscript{18}A general term referring to a URL with no visible GET parameters embedded: e.g. compare \url{http://domain.com/i.php?user=john\_doe&page=profile} (or worse) to \url{http://domain.com/user/john\_doe/profile}.

\textsuperscript{19}At minimum 4096 bytes of data per cookie (source: RFC 2109 \url{http://www.ietf.org/rfc/rfc2109.txt}, last retrieved June 5, 2010), with some setups decryption of the cookie might be faster than several database queries. Encrypting the cookie with symmetric (AES128-Rijndael, Blowfish, for older setups 3DES) algorithm is recommended to prevent cookie poisoning by malicious users.
2. Theoretical requirements

**Figure 2.9:** PHP session id generator; tv: struct containing server time from `gettimeofday()`, remote_address: public IP of a client, `php_combined_lcg`: pseudo-random number, seed: server time

```c
sprintf(&buf, 0, "%.15s%ld%ld %0.8F", 1
remote_addr ? remote_addr : "", 2
tv.tv_sec, (long int)tv.tv_usec, 3
php_combined_lcg(TSRMLS_F) * 10 4
);
5
6
7switch (PS(hash_func)) {
8    case PS_HASH_FUNC_MD5:
9        PHP_MD5Init(&md5_context);
10        PHP_MD5Update(&md5_context, (unsigned char *)buf, strlen(buf));
11        digest_len = 16;
12        break;
13
Source: PHP source code, `ext/session/session.c`, l. 370+, PHP 5.3.2, Debian patches (5.3.2-1).
```

**Figure 2.10:** Django session identifier generator

```python
def _get_new_session_key(self):
    """Returns session key that isn’t being used."""
    # The random module is seeded when this Apache child is created
    # Use settings.SECRET_KEY as added salt.
    try:
        pid = os.getpid()
    ...
    while 1:
        session_key = md5_constructor("%s%s%s%s" % (randrange(0, MAX_SESSION_KEY), pid, time.time(), settings.SECRET_KEY)).hexdigest()
        if not self.exists(session_key):
            break
    return session_key
```

Source: Django framework source code, `contrib/sessions/backends/base.py`, l. 131+, 1.2 beta1-1 (Debian).

In case of Django, several differences can be spotted. As a source of “randomness” per user, process ID\(^{20}\) is used. Also, in comparison to PHP, any attack\(^{21}\) trying to predict the state of random number generator (`php_combined_lcg()`) is mitigated by a per site unique `SECRET_KEY` (50 alphanumerical characters generated from a set of 26 lower case letters, numbers and 14 “special” characters) – this string is, compared to the state of `php_combined_lcg()`, unpredictable even

\(^{20}\) Might be problematic with some setups, where one process serves multiple requests or clients.

\(^{21}\) Short description: [http://seclists.org/fulldisclosure/2010/Mar/519](http://seclists.org/fulldisclosure/2010/Mar/519) (last retrieved April 2, 2010), can be eliminated by applying Suhosin patches. For IPv6, even less entropy is supplied (only first six octets of the address); server time can expected to be synchronized by NTP.
for shared web hosting. Both examples use MD5 hash function, which is no longer considered cryptographically secure for most uses\(^{22}\). PHP also allows (the default is MD5) using either SHA-1\(^{23}\) or user supplied hash function.

One last problem associated with cookies remains to be discussed. Cookies can be (and often are) employed to track users (a very important task in e-commerce applications). However, some users may block cookies or delete them randomly (see Chapter 4 for more information). Such action is not important for the actual site (they expect to be logged out), but eliminates the possibility of tracking that is vital for affiliate software. Fortunately, other option is available – Local Shared Objects (LSO), supplied by Adobe Flash Player. These so called “super cookies” are not deleted with standard browser functions and have to be manually located on the disk and manually deleted. By conservative estimates, more than 75 % of visitors’ browsers\(^{24}\) support Flash, so it is a viable auxiliary option.

Figure 2.11 provides an example of such flash cookie (created with affiliate software PostAffiliate Pro). Despite the binary file with unknown structure, all the data is easily distinguishable (IP address, unix times, affiliate partners’ codes, source domain). Standard cookie with exactly the same information is saved too. Although it might be tempting to use LSO more, one should bear in mind that it is impractical in any security-related situation (due to security track of Adobe Flash and due to the difficulty of deleting flash cookies e.g. in a public internet café – LSO’s are buried deep inside hidden directories\(^{25}\)) and also such cookies may not be legal in some countries (or certain uses of such cookies, e.g. tracking visitors through multiple sites).

### 2.2.3 User authentication and accounts

Another important issue, closely related to sessions, is user authentication, or linking users with their respective accounts. The form of authentication with the lowest barriers for users is employing a combination of user name (identity) and password (for proving the ownership of identity during authentication) and an HTML form

\(^{22}\)See [25]; with sufficient salt, MD5 can still be used for cookies.

\(^{23}\)Despite the choice, many PHP applications may not support session identifiers longer than 128 bits (16 characters) – e.g. in Drupal with plugin Ubercart 1.x, visitors’ shopping carts identifiers – essentially PHPSESSID (session tokens) – are stored in a database field with a size of 16 characters (128 bits).

\(^{24}\)http://www.adobe.com/products/player_census/npd/ (last retrieved April 2, 2010; author’s statistics)

\(^{25}\)For Linux: 
- .macromedia/Flash_Player, for Windows: %APPDATA%\Macromedia\FlashPlayer#SharedObjects\ (and possibly others).
Figure 2.11: PostAffiliate Pro 4.x. LSO cookie, unprintable characters discarded (binary file), anonymised.

```
{ "cl":null,"ba":null,"pb":null,"ts":1270228368,
 "rf":"H_sourcedomain.com/index.php?id=info",
 "ip":"1.2.3.4","d1":"","d2":"","ch":""}  
PAPCookie_Sale  
{"c":"f8ba37","a":"96bad",
 "ch":null} PAPCookie_LastClick  
{"cl":null,"ba":null,"pb":null,"ts":1270231610,
 "rf":"H_sourcedomain.com/index.php?id=info","ip":"1.2.3.4","d1":"","d2":"
 :"","ch":""}  
PAPCookie_Time
```

Source: Author, PAP 4.2, ~/.macromedia/Flash_Player/#SharedObjects/()/domain.cz.

(and HTTP GET/POST request) to submit the pair. After successful authentication, the user is handed a session token (in a form of cookie, GET parameter or via hidden form fields). General requirements for authentication (unique user names, unambiguity of passwords) apply.

It should be noted that for most B2C sites, no real, off-line confirmation of users’ claimed identity is performed (due to the nature of e-commerce and willingness of users to hand out information) and only one factor (password) is available, hence the actual process of creating user accounts (registration) should be highly simplified. A common solution is to create an account during checkout process, allowing customers to choose username (or use email address as username) and password. The dual nature of e-commerce security is easily manifested during registration. From the security point of view, password rules (minimum length, special characters) must be enforced. However, from the business point of view, the site must track customers and therefore the effort to remember (or retrieve) password should be much lower than the time needed to fill in customer details during checkout. Other such requirement is the validity of email addresses – security dictates verifying ownership (by sending email with unique link), but the possible delay (minutes

---

26Other options could include variants of HTTP Authentication, see IETF RFC 2617 (HTTP Authentication: Basic and Digest Access Authentication), http://www.ietf.org/rfc/rfc2617.txt (last retrieved April 4, 2010), PKI, multichannel authentication.

27Handling sessions with hidden form fields prevents certain attacks such as stealing cookies, but is often only used with critical applications (most notably, internet banking) due to complicated code (POST request for any page, including links, session lasting only until the user closes the browser window) and requirement of JavaScript. The sessions in forms may be, to some extent, handled automatically by e.g. ASP. Hidden form fields as only authentication measure are unsuitable for e-commerce (typical buyer adds products to cart, waits several days and then finishes the checkout process (source: author’s data). For high-profile targets, multiple tokens may be handed (via cookies, hidden form fields and custom JavaScript variables – see Amazon.com checkout process).

28See Chapter 4 for the list.
to hours) also means that the customer may either decide not to buy the product or buy it elsewhere.

The contradiction is also present during login. The number of login trials must be limited (to prevent password guessing), but it also must allow the customers to guess their passwords (especially in case of casual buyers). The usual solution is adding a delay (after submitting the password, but before the actual authentication – see Subsection 3.3.7) to slow down the possible attacker and limit the number of trials at value unachievable by human customer, but still efficient against determined attacker\textsuperscript{29}. Information leaks should also be taken into account – a convenient function making difference between wrong username and wrong password could be used to check whether a certain person has used the site\textsuperscript{30}. The user account itself is only as secure as is the user's email account (password retrieval).

An important issue is storing user passwords. If the attackers can access users' credentials, then they can also most probably access any other part of the e-commerce site and so it may appear to be useless to additionally protect passwords (in an ideal case – concealing passwords from staff is another reason). However, it is probable (see Section 4.1) that many users will use the same password for their e-commerce and email account, and so the passwords must be protected in some way (both because of legal compliance and to prevent spreading the news about lost user accounts\textsuperscript{31}). The usual procedure are one-way hash (message digest) functions, with salt comprising of information unique for customer (e.g. username) or a random salt (stored together with password). From the point of security, it is preferable to also add salt unique for the site, stored in a file outside of the database (in case of a database compromise). As for cryptographically secure hash functions, SHA-2 family is recommended\textsuperscript{32} and very common MD5 should be considered broken\textsuperscript{33}.

The last problem related to user accounts is privilege separation. It must be possible to limit users (or user groups) with high granularity both inside and outside the e-commerce application. Such requirement is both a security necessity and also a valuable business option (e.g. assigning discounts to different users). A nice definition can be found in ISO/IEC 17799 (Figure 2.12); a general framework in [40].

\textsuperscript{29}Other, more complicated rules can be enforced, such as no more than 100 logins for 10 different accounts per IP per hour with no more than 100 failed logins.

\textsuperscript{30}Such vulnerability creates a legal liability in case of adult toys store or in case of a site selling luxurious products.

\textsuperscript{31}The risk of noticing abused email account (used daily) by customers is far higher than the risk of noticing a security incident of an e-commerce site.

\textsuperscript{32}[25]; it is still possible to use SHA-1, although its use for newly designed systems is questionable.

\textsuperscript{33}Extracting weak passwords with rainbow tables and other time-memory trade-off attacks is computationally feasible. For MD5 security, see for example [33], [5] and [25].
2.12.2 Privilege management … The allocation and use of privileges should be restricted and controlled. (…) The following steps should be considered:

a) the access privileges associated with each system product, e.g. operating system, database management system and each application, and the users to which they need to be allocated should be identified;

b) privileges should be allocated to users on a need-to-use basis and on an event-by-event basis in line with the access control policy (11.1.1), i.e. the minimum requirement for their functional role only when needed;

c) an authorization process and a record of all privileges allocated should be maintained. Privileges should not be granted until the authorization process is complete;

d) the development and use of system routines should be promoted to avoid the need to grant privileges to users; (…) 

f) privileges should be assigned to a different user ID from those used for normal business use.

Source: [17, pg. 62, 11.2.2 ].

Note that in case of small e-commerce sites, most of IS/ICT roles (e.g. system administrator) will be outsourced and bought as a service (for example server administration). Also note that the privilege separation does not only cover the actual application, but access to data in general – therefore it should be possible to e.g. alter or configure the application without being able to access customer data.

2.2.4 Input sanitation

By definition, large parts of any e-commerce application are exposed to any request from any source and there is no way to ensure correctness of data sent to the server from client. This implies that a set of strong validation rules should be employed upon receiving any data from client to check whether it is legitimate and not harmful. Unfortunately, no rigorous type enforcement is possible (e.g. converting input internally to integers) as most data contains plain text.

Unsanitised input is also a large source of e-commerce application vulnerabilities in terms of willingful attacks and may be used to directly attack the application itself (e.g. SQL injection) or to attack the client (mostly through scripting languages, but also through drive-by-download). The basic problem is identifying malicious input. While for the application (e.g. PHP scripts) the input may be harmless, the attacker can access underlying systems (database, operating system) through

\[\text{eval()}, \text{call_user_func()}, \text{call_user_func_array()}, \text{call_user_method()}, \text{call_user_method_array()}, \text{create_function()}.\]

See Chapter 3 for verification of the claim. Except if some parts of the input are evaluated as code.
the application. An effective way of protecting the application (and underlying systems) is to escape any input as soon as it is received\textsuperscript{36}. However, such protection does not prevent the attacker from storing and later displaying arbitrary data to clients (or performing second order SQL injection) – a second filter should be employed to classify input into categories and filter the outputted content according to the pre-set rules (e.g. allowing outputting HTML for some users while only plaintext for others) with the low risk behaviour (plaintext) as default.

A vigorous cleaning of input must take place in case of directly executing other programs (outside the application\textsuperscript{37}) or if any files are opened or included\textsuperscript{38} (path traversal attacks).

When we start considering client computers, the situation gets even more difficult. Filtering SQL injection attacks is relatively simple\textsuperscript{39}, filtering all possible attack vectors to the client is complicated. At first glance, it seems that allowing input from only trusted sources (employees) is sufficient – however, in modern applications, even input from users is stored and also, users’ computers allow running a relatively high amount of code (JavaScript, Java applets, Adobe Flash, Microsoft Silverlight and ActiveX content to name at least few). Browser behaviour is erratic (older versions of MS Internet Explorer ignoring MIME type image/jpeg if the content can be interpreted as text/html), input can be delivered through multiple ways (“classical” HTTP request, AJAX, Flash application) and can often be handled inconsistently. Enforcing file types is achievable by checking file headers, but cleaning all input from any JavaScript code is much harder\textsuperscript{40}.

There are many ways in which an attacker can abuse input filtering vulnerabilities in applications. To name the most “interesting” ones, SQL injection may be used to access user data (or at least enumerate them) and XSS is gaining popularity. XSS is divided into two categories: reflected and persistent (stored; terminology from \textsuperscript{[6]}).

Even though code in Figure 2.13 appears “mostly harmless”, it can be easily abused (requires very limited knowledge) and to some users makes the actual site appear to be “hacked”\textsuperscript{41}. A persistent XSS is usually achieved by storing some

\begin{itemize}
\item Blueprint CMS uses this approach.
\item PHP functions exec, passthru, popen, proc_open, shell_exec, system; often disabled to some extent in php.ini (directive disable_functions). Note: the list of disabled functions is usually much longer; see Appendix D.
\item A common attack is to force the application to include and run code from a different server. In PHP this can be prevented by setting allow_url_fopen to off.
\item Generally, removing or escaping (or converting to HTML entities) any characters interpreted by the database engines as delimiters, e.g. ‘\’’, or by utilising stored procedures.
\item Removing certain characters (<, >) is not enough (they can be disguised through the use of e.g. different encodings), the only sufficient protection is to convert all “special” characters (not matching [a-zA-Z0-9]) to HTML entities.
\item A reflected XSS vulnerability in a site of one of larger Czech political parties, CSSD, has been employed for a defacement of the site with a video and re-
content on the site and is slightly more efficient (requires no user action to run).

Both XSS options allow spreading malware, redirecting users or stealing cookies (a stored XSS overcomes same origin policy of the browser) and thus a vulnerable field inside a checkout process could be used to retrieve passwords of administrators or staff (or at least for session hijacking\(^{42}\); JavaScript can record keystrokes, attack auto-complete features of any browser etc.). So far, it may seem that a threat comes only from the “outside”. However, insufficient input sanitation is a reliable source of data for unloyal employees too. XSS also removes any trust from transactions – a user could be forced to finish a transaction (payment, order) without consent.

A specific threat appears from third party applications (essentially uncontrolled input sent to the user). A great deal of vigilance should be applied with regard to foreign code and the least damaging option should be used to include it. An example of such application is invaluable Google Analytics\(^{43}\), another could be a popular Facebook panel. Instead of directly adding any code to the site, iframe is the preferable, slightly more secure option. If any banners are exchanged, all images must be stored in a controlled environment, never loaded from a foreign site. Third party applications may cause data leaks – a prime example is a visitors’ counter TopList, which by default allows accessing important data (number of daily visitors and comparison to the past) even if displaying the statistics on the website is disabled.

It is arguable whether integration with certain services (for example PPC systems like Google AdWords) violates any data protection directives. Theoretically, if conversions are tracked, especially Google might detect whether a certain visitor bought any items and link that information to respective user accounts. Such information may not be important per se, but could be abused if the e-commerce sites sells a narrow selection of goods (e.g. religious books). The responsibility lies on the controller of private data, which means owner of the site.

---

\(^{42}\)Preventable by setting cookies HTTPonly.

\(^{43}\)Apart from input, user privacy and business data are in danger too – essentially even through a small banner, one can easily measure site conversion rate and spread malware.

---

2. Theoretical requirements

Figure 2.13: Reflected XSS example

\[
\text{http://domain.com/error.php?error_message=\\textlt;script\\textgt>alert('XSS');\\textlt;/script\\textgt;}
\]

Source: Author; hypothetical application directly outputs request parameter `error_message` as the error which is then is executed by the browser.

Another type attack is a redirect – mostly through unsanitised redirect URL (and no enforcing of local URLs) in a login form. It may be used to redirect the user to another site (scam, malware download) after seemingly “successful” action (the first page seems to be valid and trustworthy). Similar attack is achievable by injecting HTTP headers [6, pg. 434] (which can also be used to session fixation, described earlier) and HTTP response splitting (omitted, as such threat is not significant to small e-commerce sites).

Figure 2.14: Redirect attack


Source: Author; hypothetical application, vulnerable redirect (redirect_to is destination site).

Request forgery, another attack, has two types – on-site (OSRF) and cross site (CSRF). OSRF forces user’s browser to perform unwanted action through stored XSS vulnerability\(^{44}\) or by submitting requests e.g. via `<img src=''>\(^{45}\) targets. CSRF is exploited by forcing users to unwittingly submit requests from one site to another (either with JavaScript or other means). The result of such attacks varies from adding comments spam to adding admin privileges to the adversary. An effective protection, to some extent, is offered by requiring a randomly generated token (tied to IP address and time) to be submitted with every form in the application and referer checking. Thus any user has to visit the page with form in order to be able to submit it. In Figure 2.15, an example (generated by Django framework) is presented – CSRF protection is handled completely automatically (in Django through middleware). Tokens can also be utilised to prevent forces browsing – attacker skipping several parts of multiple page forms, especially validation, and sending the request directly (in e-commerce – avoiding payment in the checkout process).

\(^{44}\)For example famous Samy MySpace worm.
\(^{45}\)[6, pg. 441]; such flaws emphasize the necessity to differentiate between POST and GET variables in requests.
2. Theoretical requirements

Figure 2.15: CSRF protection

```html
<form method="post" action="/contact/">
  <div style='display:none'>
    <input type='hidden' name='csrfmiddlewaretoken' value='27da0cf30cea72d7f86b07a605d2fe0' />
  </div>
</form>
```

Source: Author; CSRF protection, Django framework.

2.2.5 Encryption

So far, a crucial requirement of e-commerce security has been ignored – encryption (protection against eavesdropping). While from the point of security, any communication\(^\text{46}\) (from DNS to HTTP) should be encrypted, or at least signed to ensure message integrity, due to costs and performance requirements\(^\text{47}\), such scheme is rarely implemented (see a small benchmark in Appendix A).

For DNS, DNSSec can be set-up, but it’s usage is still low (but growing). For HTTP communication, SSL should be used (either SSL or TLS, depending on browser support) with approved ciphers (e.g. AES, 3DES in appropriate mode) for any important data – login information, checkout process, etc. [23]\(^\text{48}\) A commercially-signed certificate is a must – for self-signed certificates or certificates with untrusted root CA, browsers display a rather menacing warning\(^\text{49}\). The application should distinguish between data and never send any both through secure and insecure channel. This means that for example during authentication, cookies must be regenerated (as the already assigned session token has been possibly compromised). A dual system (“real” token for accessing private data, checkout etc. and a “limited access” for cart functionality and personalisation), used e.g. at Amazon, provides a nice solution. A good practice is to secure even the login page itself, so that end users can verify validity of the SSL certificate (and assume that the form will be submitted over a secure channel).

Even though the focus of the thesis are applications, not underlying operating

\(^{46}\)At the scope of small e-commerce application.

\(^{47}\)Not only encrypting the content, but also generating enough entropy for RNG.

\(^{48}\)Also referencing cryptographic strength, cypher usage.

\(^{49}\)Unfortunately, the warning makes connection with “some” encryption (but no verification) appear to be less secure than connection with no encryption at all. Also note that root certificates are subject to browser policies, so many local CAs are not included (as is the case of the three Czech authorities recognised by the Ministry of Interior, but not by the majority of browser manufactures) and conversely certain undesirable root certificates may be in the browser (e.g. a hypothetical CA controlled by the Chinese government – even a single faux authority is enough to compromise the chain). See Chapter 4 for user reactions to an invalid certificate.
systems, the author would like to stress that a cryptographically strong random number generator is required for successful encryption of the traffic.\textsuperscript{50}

The only real weak points that cannot be easily secured are other “channels” like email – passwords are commonly sent over email, which is presumably unencrypted communication at all times except for the last “mile” (client to email server). One could simply disregard the threat as insignificant, but in author’s experience, the threat proves to be very real, despite merely anecdotal evidence. Generally, data is considered to be most vulnerable to eavesdropping at the local ISP level (especially with wi-fi connection), but such threat is dangerous for the client, not the e-commerce site. A more devastating accident (except for voluntary interception of data at the data centre) can happen in case of malfunction of networking equipment. Example: the author once detected unreasonably high incoming data (see Appendix B) and after discarding any legitimate sources, a true source was discovered – a Czech “RapidShare-like” server’s complete traffic was sent to the author’s server, including usernames (email addresses) and passwords in plaintext (that could be captured at the speed of about 30 pet hour through tcpflow and crude filtering). For e-commerce site, such data leak is undetectable and could endanger the whole business, as the potential attacker has access to aggregated data from many users, not just one.

A minor issue prevents employing more SSL secured domains – in shared environments with name-based virtual hosts, the server has no way to determine packet destination (and so use the correct certificate), so a unique IPv4 address (becoming a scarce good) has to be used for each site (TLS allows specifying domain name outside of the encrypted part of the packet, but its browser support is limited).

Encrypting communication is not only important for “customer” to server communication, but for “server” to “server” communication too – e.g. affiliate software checking licences should encrypt all data and also verify (e.g. through PKI) that the source is the actual server. See example of intercepted unencrypted communication of PostAffiliate Pro, 4.x.x.x in Appendix C. The same software explicitly disables any certificate authority validation:

\textbf{2.2.6 Logs, data retention, backups and others}

The last part is dealing with various “remaining” security requirements. One of them is maintaining a “good SEO profile”, which means cleaning the site from spam

\footnote{See \url{http://wiki.debian.org/SSLkeys} (last retrieved June 5, 2010) for more information about weak (broken) example of RNG in Debian OpenSSL package. Such vulnerabilities might exist undetected in many more systems.} \footnote{bagruj.cz}
and also assigning an employee to monitor content of the site (to prevent phishing, copy cats, content scrapping etc.) Logs should be kept in accordance with “acquis communautaire” ([12]) and local data retention laws, and backups plan should be established (but both of these requirements are focused towards service providers, not the actual e-commerce application).

One last issue before the thesis continues with case studies – secure deletion of content. In e-commerce applications, no content should ever be deleted. Merely disabling the content (and saving differences between versions of the same content) enables later checking and faster recovery.

Hiding parts of the site without proper access controls is not recommended, as most search engines will eventually find the hidden files (and robots.txt can be easily used by the attacker to get a list of “hidden” – and potentially vulnerable – interfaces).

A full list of remaining issues that are outside of the focus of the thesis can be found in [18] and other computer security frameworks.
Chapter 3

Security of selected open-source e-commerce applications

To accomplish the goal of the thesis and to study vulnerabilities in “real-life” applications, open-source software was selected for the assessment, as it allows analysing the source code, does not bear additional financial costs and full disclosure of discovered vulnerabilities is possible. To limit the scope, only PHP applications are considered (due to author’s familiarity with the language; see thesis limitations in Section 1.3).

An exploratory study has been conducted by the author to determine whether viable data (about attackers and possible attacks) could be collected through a honeypot with multiple versions of common open-source software at the disposal of the author. The approach was selected because the current writer could not assure that a full-scale honeypot would effectively attract enough attackers, so a limited test took place first.

Two sites were tracked (via server logs). First was a long-running e-commerce site with Zen Cart 1.3.8, which is an example of a small site: technically unskilled owner, sole ownership, core-business = furniture, mainly kitchen equipment, old, unupdated software, well indexed (206 pages\(^1\), covering all products), well linked (roughly 6670 inbound links\(^2\)). The version of Zen Cart installed (1.3.8) is outdated, with numerous security bugs\(^3\), including XSS in the administration interface\(^4\), SQL injection\(^5\), remote code execution and information leaks. As for support, patches are distributed through forum posts that contain required changes in the code, no pub-

\(^1\)Search query site:domain.cz in google.com as of April 25, 2010.


\(^5\)/includes/classes/shopping_cart.php, line 307.
lic mailing list is established and change logs are not maintained. Security through obscurity is a recommended practice (renaming folders to hide administrators’ interface\(^6\)). Presence of Zen Cart is advertised in templates and sites running the software can be searched for in Google.

On average, three different (“successful”) attacks were mounted daily against the application using pre-packed exploits\(^7\). The vulnerability (unsanitised input) theoretically would have allowed the attacker to upload an arbitrary file. Due to server configuration – directories owned by a user (domain_cz_usr) different from the user running Apache 2 (www-data) and limited PHP functionality (disabled functions), all attacks were unsuccessful in the end (attackers could not access the shell and compromise the server).

Note: the site is not maintained and not updated solely by the decision of the owner.

The second test site was running Drupal (5.20) with UberCart e-commerce module (1.6), various stock (Views, Image, CCK) and some customized (VAT, invoices, CRM and reservation systems integration) modules. The setup can be viewed as the opposite of Zen Cart (no security through obscurity, public disclosure at http://drupal.org/security and mailing list, verbose change logs etc.) Despite the security track, the current setup was vulnerable to several XSS attacks (no updates were performed due to customization of the software and anticipated changes to a different platform). No custom-tailored attacks were recorded, only generic SQL injection testing and automatic searches for vulnerable software (Zen Cart, OsCommerce) – “low-hanging fruit”.

In light of exploratory results and after a search of bugs in common open-source e-commerce packages, the author decided to not experiment with a full scale honey-pot test of many e-commerce systems (as most of the exploits can be prevented by a tighter server security policy, and therefore any successful “hacker” in an “academic setup” probably would not be successful in the real world; and thus the results would have not been credible, as any preselected outcome could have been achieved). Instead, to study the security, an evaluation based on preselected criteria was performed against source code. Such evaluation is only made possible by using open-source applications, as closed-source, proprietary software of this scale is usually not analysed nor audited or certified against relevant standards.


3. Criteria and methodology

Criteria for evaluation were created based on requirements in Chapter 2 and divided into six categories, each containing 4 items. While some of the criteria may seem to be superficial, the author believes that all of them are equally important. Each package will be evaluated for each criterion with a boolean system of results – either it complies (1) or not (0). Each criterion is either “positive” (P, complies if some effort was taken) or “negative” (N, the author found at least two cases where the criterion had been violated).

The thesis continues with the presentation of criteria.

Input sanitation (N)

Effective sanitation: The sanitation should protect against XSS and SQL injection attacks and should be centralized.

CSRF: Cross-site request forgery protection should be automatic for all forms.

Spam protection: All public forms for anonymous users should be protected with CAPTCHA. The system should not allow sending spam to arbitrary email addresses.

MIME types: All user-supplied files should have enforced MIME types.

User accounts (N)

Privilege separation: Separate customer, merchant and administrator accounts should be supplied.

Data leaks: The software should not inform users about its internal state more than is necessary.

User actions: Users should not be able to affect the state of the site or software beyond their privileges by any means (including altering cookies, hidden fields, etc.)

Unsolicited email: It should be possible to mark a user account so as not to receive email through a single link without deleting the account. The users should be able to withdraw their consent for receiving email in the interface and with a link in any received email.
3. Security of selected open-source e-commerce applications

**General security (N)**

**Updates:** The application should have an internal updating system, alerting the administrator about new versions of the software.

**Known bugs:** The current version of application available for download should not contain any known security bugs which are more than two weeks old.

**Mailing list:** A publicly accessible mailing list for security announcements should be used.

**Plugin policy:** If the application uses plugins and if application authors host or endorse plugins for download, the state of their security should be clearly defined (e.g. whether plugins are user-submitted without any checks or not) and unambiguously communicated to the user.

**Documentation and legality (P)**

**Detailed information:** All change logs should be readily available (important if the administrator needs to check whether the update is necessary or not). Full disclosure of any security issues should be practised.

**Recommendations:** Documentation should include a section including security recommendations and should mention the general legal requirements of e-commerce applications (e.g. [11]).

**Secure installation:** The installation manual should not contain any insecure steps and should leave the software in a secure state at the end of the process (“secure by default”).

**Consent:** The checkout process should contain an explicit field for customer consent with whatever legislation is required.

**Good practice (P)**

**PHP:** No unnecessary functions and programs (e.g. shell access) should be used (PHP is a templating language after all). The application should not rely on obscure, outdated or insecure PHP settings (e.g. `register_globals`).

**Verbose exceptions:** Exceptions should be verbose and should not be silenced. Employing @ operator is not error checking.
Comments: Source code should include comments at least generally outlining the role of each file or function. The code should not contain any dubious comments (“Does this work?”).

Logging: All user actions in the application should be logged.

Private data protection (N)

SSL ready: The application should not emit any warnings when used over a secure connection.

Enforced SSL: The administrator should be able to enforce encrypted communication for login and certain other parts of the site.

Session tokens: Session tokens (e.g. cookies) should be regenerated if ever transmitted over an insecure channel (pre/post login).

Cryptography: All passwords should be stored in a cryptographically secure way.

The criteria have been compiled with regard to [7] (especially PPTM and STRIDE frameworks, pg. 90, and best practices, pg. 91), [32] (especially “Good practice”), [28], [18], [17], [31] and [37].

Installation instructions will be meticulously followed (except for the fact that the author will be using SSH and working directly on the server, instead of unpacking the software at a workstation and uploading it via FTP). SSL certificates are signed by an authority controlled by the author (CA certificates are present in the browser and provide chain of trust). MySQL user bp and database names bp_(softwarename) will be created for the test. The basic rule for evaluation is “secure by default”, which means that any security measure must be turned on after installation and basic configuration or else it is considered missing and all filtering must take place unless turned off (e.g. selectively setting which messages to protect from XSS will be evaluated as no protection at all). There are exceptions to the rule – the author will try to enable SSL, CAPTCHA and CSRF protection if possible in the configuration, and if possible, install any additional software at the same level as the e-commerce package (this is the case of e-commerce software distributed as a module/plugin of a larger CMS, where other plugins are “at the same level”; if the shopping cart is provided as a standalone version, no additional software will be searched for).

All file operations were performed by system user bp with no extended privileges.
3. Security of selected open-source e-commerce applications

Figure 3.1: Database setup

```sql
CREATE DATABASE 'bp_software_name' DEFAULT CHARACTER SET utf8 COLLATE utf8_unicode_ci;

CREATE USER 'bp\0'@'localhost' IDENTIFIED BY '****';

GRANT USAGE ON *.* TO 'bp\0'@'localhost' IDENTIFIED BY '****' WITH MAX QUERIES_PER_HOUR 0 MAX_CONNECTIONS_PER_HOUR 0 MAX_UPDATES_PER_HOUR 0 MAX_USER_CONNECTIONS 0;

GRANT ALL PRIVILEGES ON 'bp\%'.* TO 'bp\0'@'localhost';
```

Source: Author, queries generated by PhpMyAdmin

3.2 Software selection

The selection of software packages is highly subjective. By the scope of the thesis, the limitation is e-commerce software written in PHP\(^8\) that can be run on Linux servers. The versions are limited to the last current version, alpha or beta quality releases are not considered\(^9\). To yet more streamline the selection, the cart should have (as of April 20, 2010) a stable release not older then 12 months\(^10\)), must support multilingual features and should be released under an approved\(^11\) open-source license\(^12\).

Enumerating all possible shopping cart (e-commerce) software solutions is virtually impossible, so several sources have been used to select possible participants of the test\(^13\). To measure popularity of each package (to remove little used solutions), an indirect method was selected (as most suppliers do not provide credible information on the exact amount of installations) to compare the systems (i.e. get an estimate, of which software is more popular). The popularity value in the following table means the number of times the software was mentioned on a page indexed by Google, excluding the software’s site\(^14\). The evaluation was conducted during March

\(^8\)Disqualifies nopcommerce.com.

\(^9\)Disqualifies OsCommerce (oscommerce.com), supplying only 2.2 Release Candidate 2a and v3.0 Alpha 5, released in March 2009, Batavi (batavi.org), supplying only 0.92 beta as of January 27, 2010.


\(^11\)http://www.opensource.org/licenses/alphabetical (last retrieved April 25, 2010); license must be open-source, not “free as in beer”, because such license may restrict code analysis.

\(^12\)Disqualifies StoreSprite (storesprite.com), Cubecart (cubecart.com) and various others.


\(^14\)Search query software_name-site:software_site.com.
and April and results were checked at the end of April for any changes (in case of a new version, the source code was compared via `diff` utility).
### Table 3.1: Software selection overview

<table>
<thead>
<tr>
<th>Name</th>
<th>License</th>
<th>Version</th>
<th>Release date</th>
<th>Website</th>
<th>Perceived popularity in the media (thousands)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Magento</td>
<td>OSL 3.0</td>
<td>1.4.0.1</td>
<td>Feb 19, 2010</td>
<td>magentocommerce.com</td>
<td>7250</td>
</tr>
<tr>
<td>OpenCart</td>
<td>GPLv3</td>
<td>1.4.7</td>
<td>Apr 18, 2010</td>
<td>opencart.com</td>
<td>563</td>
</tr>
<tr>
<td>Oxid eshop community edition</td>
<td>GPLv3</td>
<td>4.3.1</td>
<td>Apr 18, 2010</td>
<td>oxid-esales.com</td>
<td>3720</td>
</tr>
<tr>
<td>Prestashop</td>
<td>OSL 3.0</td>
<td>1.2.5</td>
<td>Oct 29, 2009</td>
<td>prestashop.com</td>
<td>2250</td>
</tr>
<tr>
<td>Ubercart (Drupal)</td>
<td>GPLv2</td>
<td>2.2</td>
<td>Nov 19, 2009</td>
<td>ubercart.org</td>
<td>7460</td>
</tr>
<tr>
<td>VirtueMart (Joomla)</td>
<td>GPLv2</td>
<td>1.1.4</td>
<td>Oct 18, 2009</td>
<td>virtuemart.net</td>
<td>18900</td>
</tr>
<tr>
<td>Zen Cart</td>
<td>GPLv2</td>
<td>1.3.9a</td>
<td>Apr 19, 2010</td>
<td>zen-cart.com</td>
<td>17800</td>
</tr>
</tbody>
</table>

*Source:* Author, software's web pages as of April 20, 2010. Oxid release date is an estimate (latest possible, due to multiple minor patches available); the searched term was “oxid e-commerce” (for "oxid community edition", 164 thousand results). In brackets, underlying CMS (if any).
The author has to note that *none* of the sites provided *any* checksums of the files and none of the sites (and the downloads), including community editions of commercially available software, fully supported SSL/TLS (mainly invalid certificates or HTTP-only communication), so the integrity of the downloaded software cannot be validated by any means.

The test server configuration is listed in Appendix D.

### 3.3 Assessment

First, the results of the assessment performed by the author are presented. Then, specific notes regarding each package are mentioned.
### Table 3.2: Assessment results

<table>
<thead>
<tr>
<th></th>
<th>Magento</th>
<th>OpenCart</th>
<th>Oxid</th>
<th>Prestashop</th>
<th>Ubercart</th>
<th>VirtueMart</th>
<th>Zen Cart</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input sanitation</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Effective sanitation</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CSRF</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Spam protection</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>MIME types</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>User accounts</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Privilege separation</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Data leaks</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>User actions</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Unsolicited email</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td><strong>General security</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Updates</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Known bugs</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Mailing list</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Plugin policy</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>Documentation and legal</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Detailed information</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Recommendations</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Secure installation</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Consent</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td><strong>Good practice</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PHP</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Verbose exceptions</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Comments</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Logging</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td><strong>Private data protection</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSL ready</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Enforced SSL</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Session tokens</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Cryptography</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td><strong>Totals</strong></td>
<td>15</td>
<td>9</td>
<td>17</td>
<td>9</td>
<td>21</td>
<td>14</td>
<td>12</td>
</tr>
</tbody>
</table>

*Source: Author.*
3. Security of selected open-source e-commerce applications

3.3.1 Magento

Magento is probably the most extensive software evaluated (due to commercial support and commercial versions). With default configuration, the software would not install (it attempts to write to shared directory /tmp, which is prohibited by open_basedir directive), therefore the application was altered by the author as follows to store temporary data inside accessible directory:

```
public function getSysTmpDir()
{
  //return sys_get_temp_dir();
  return $_SERVER[ 'DOCUMENT_ROOT' ] . 'magento/shared_tmp/';
}
```

Source: Author, app/code/core/Mage/Core/Model/Config/Options.php, l. 133.

Several temporary files were created, and with the structure and naming, if placed in /tmp in a shared environment, they would both prohibit other users from installing Magento and also allow them to access potentially sensitive data (cache, sessions).

```
$ ls -lh shared_tmp/magento/var/
total 8.0K
drwxrwxrwx 18 www-data www-data 4.0K 2010-05-01 23:18 cache
drwxrwxrwx  2 www-data www-data 4.0K 2010-05-01 23:15 session
```

Source: Author.

During installation, password complexity is enforced. However, the form is not submitted over HTTPS (the same holds for a secure key for credit card information encryption). A string with exact version of the software is shown on the home page (“Help Us to Keep Magento Healthy - Report All Bugs (ver. 1.4.0.1)”), which allows attackers to rapidly find and exploit vulnerable versions.

For storing passwords, MD5 is used with 2 characters long random salt (unique for each password). To the authors knowledge, the added complexity should prevent rainbow tables attack to a limited extent (the salt is concatenated with the password) especially in case of weak passwords\textsuperscript{16}, but for strong passwords, the added complexity is negligible. Usernames are case-insensitive (by default).

\textsuperscript{16}Against rainbow tables and time-memory trade-off attacks.
3. Security of selected open-source e-commerce applications

No check is performed against session tokens, although fixating user agent, IP address and proxy IP address can be configured (disabled by default; see app/code/core/Mage/Core/Model/Session/Abstract/Varien.php, l. 350).

All logging is disabled by default, logging user actions is possible only in enterprise version. Emails generated through “Recommend to a friend” function are not logged nor rate-limited.

HTML escaping is present, but not consistent (it has to be turned on for each string). Previous versions were vulnerable to XSS attacks due to poor escaping. CSRF protection is only used for administrator’s forms.

Overall, the software is reasonably secure, especially due to a very clean and understandable design (despite the size of the sources). The author must note that some of the deficiencies are not present in the purchased versions of software (mailing list and security guides are included in the paid support package).

3.3.2 OpenCart

Opencart was a contestant with probably the smallest install base (based on Table 4.5). No problems were encountered during installation (apart from inconsistency between manual and the installation script as to what directories must be writeable). Deleting installation directory was recommended, but not enforced by the software. All passwords during installation were sent in plaintext. Several world-writeable files and directories were left spread all over with no access restrictions.

OpenCart is not SSL compatible – even though SSL can be enforced for customer logins, it does not affect administrator’s login form. No session tokens regeneration has been detected.

Passwords are stored in form of MD5 hashes with no salt. HTML escaping is effective, centralized and on by default; for SQL, it is effective, but the sanitizing function must be called for each query by hand. Spam protection is performed by CAPTCHA image, but with a larger user base, such image would prove to be ineffective (constructed only by altering colours)\(^{17}\).

There were several community contributed plugins, but the software authors did not state whether those plugins underwent any scrutiny. PHP library dompdf 0.5.1 is included in the default installation, even though an announcement by OpenCart programmers recommends deleting the whole folder (unused in OpenCart at the time) due to security vulnerabilities.

No security recommendations nor a mailing list exist. The software does not

\(^{17}\)See e.g. [41] for CAPTCHA development.
check for updates. Source code comments are sparse and error reporting is suppressed by the @ operator in multiple files.

Overall, OpenCart provides some security, but for its intended user base (very simple e-commerce sites), it significantly lacks functions helping “less-computer-literate” users (e.g. security documentation).

### 3.3.3 Oxid e-shop community edition

Oxid eShop community edition is the second representative of a software with “commercial” background and in number of lines and possibilities seems to be slightly smaller than Magento and slightly older (with regard to core technologies, e.g. widespread use of JavaScript in Magento).

The installation did not use SSL, however, it contained a detailed guide and left the system in a secure state. No alterations of the source code were required. SSL can be easily enforced, but sessions are not regenerated when switching between encrypted and unencrypted requests. CSRF is complete and automatic. No security guide was found by the author.

Passwords are stored in form of MD5 hashes with 32 digits number as salt, based on time (microtime) and session identification (core/oxutilsobject.php, l. 194), which can be defined as secure for intended purposes. Source code comments were very coherent and detailed. SQL was escaped, but not consistently (e.g. part of the system used a database abstraction, part tried to escape variables directly). User input was sanitized consistently throughout the system (automatically in templates – Smarty templating engine).

CAPTCHA was present, but consisted of fixed font letters with no obfuscation, which is the least complicated text to read through OCR. The newsletter did not contain any unsubscribe link. The company actively supports community development, but a coherent plugin policy (projects.oxidforge.org) was present.

Oxid eShop can be considered secure, if compared to other, similar software.

### 3.3.4 Prestashop

Prestashop is a mid-range “small e-commerce software”, with some corporate support available. The installation process enforced password complexity, but was accomplished over insecure channel. At the end of the process, deleting installation script and renaming folder with administrator’s interface was required – this has been recognized by the author as “security through obscurity”, because the interface is protected by a password and hiding it should not improve security. The
documentation contains some recommendations regarding server security, but no guide for store owners.

HTTPS can be forced for site front end, however, the default page contains some files transferred over HTTP (which means that the browser emits a warning). Administrator’s interface is accessible with no encryption despite the configuration. Session tokens are not regenerated.

The process of creating customer’s account does not contain any informed consent regarding private data and incoming emails have no link to remove email address. A potential “DoS-like” vulnerability and a data leak exists, as the password reset function does not require a confirmation, directly resets the password and does not allow resetting it again for 6 hours.

Cookies are encrypted with Blowfish with fixed initialization vector, passwords are stored as MD5 hashes with per-store fixed salt (56 alphanumeric characters). No spam protection is present (contact form, registration). CSRF protection is disabled by default.

Prestashop has a wide community providing plugins, and a clear policy is stated on the web page: “This module or theme is provided on an “as is” basis and without warranty, either express or implied, including, without limitation, the warranties of non-infringement, merchantability or fitness for a particular purpose. The entire risk as to the quality of the original work is with you.” (source: prestastore.com, last retrieved April 28, 2010).

Exceptions are suppressed in the code via @ operator, especially in modules (included in the official installation package). Templates are not used extensively and parts of the code contain a mix of PHP and HTML (with per-case sanitation and HTML escaping), while other parts of the code use Smarty templating engine and built-in escaping.

### 3.3.5 Ubercart

Ubercart is the most “Lego-like” software from the selection, as it is based on Drupal CMS and in the default installation, only the basic system is available. To mitigate that, Simplenews, Secure Pages and Secure Pages Hijack Prevention modules were installed (see assessment methodology for the reasons).

The installation was secure (although passwords were transferred in plaintext). An extensive “Drupal handbook”, including sections about secure programming and securing the site\(^\text{18}\), is available. Drupal can be considered more a framework

\(^{18}\text{http://drupal.org/node/360052 (last retrieved April 28, 2010).}
3. Security of selected open-source e-commerce applications

rather than a CMS, and extensive logging, content revisions tracking etc. is already provided.

MD5 is used for passwords with no salt. CSRF is automatically deployed if the programmer employs internal forms builder. All input is effectively sanitized (database abstraction, text sanitation differentiable by user roles and content types). Some of the modules are of questionable quality, but it is clearly stated that they are community-contributed. Issues (bugs) are tracked both for core and for modules. Mailing list is available and Drupal has even established a security team.

Ubercart leveraged the Drupal base, which offers a very clean and stable platform for programming. The only problems encountered were inadequate password hashing and a lack of “legal” support (opt-out choices for mailing lists, informed consent during checkout and so on – however, same lack of “localisation” to European environment is present in e.g. VAT modules).

3.3.6 VirtueMart

VirtueMart is similar to Drupal (in a way that it is a CMS plugin-extension too). However, from the users’ point of view, it comes from the “other end”, comparable to a completely-assembled Lego where Drupal (with Ubercart) provides just the bare minimum (i.e. enabling plugins and configuring versus disabling plugins and changing configuration).

The author used a bundled installation (Joomla installation pack with VirtueMart already included) and for certain specific parts, a separate source code was compared. Installation procedure is not transferred over encrypted channel. Documentation is detailed, including a security check list\textsuperscript{19}. Installation guide and the actual installation process are not always up-to-date. After installation, several files were missing which caused PHP to display errors (the installation guide recommends disabling displaying errors and warnings).

Several “oddities” were encountered – credit card numbers are stored (but encrypted) by default even if a third party payment gateway is active. Logging is disabled and after being enabled, reported errors (write permissions\textsuperscript{20}). With (and only with) URL rewriting (\texttt{mod_rewrite}) active, a quite unique system of protection against a small percentage of common attacks XSS and scans was run; such system is of course completely ineffective against POST requests (see Figure 3.4). SSL encryption can be enforced for customers’ login, but not for the administrator (but a

\textsuperscript{19}\url{http://docs.joomla.org/Category:Security_Checklist} (last retrieved April 28, 2010); the author believes VirtueMart documentation should include a link to Joomla security check list (which it does not).

\textsuperscript{20}Technically, the form requested a filename, not a file path and did not check whether the file exists or is writeable with server user permissions.
large padlock is displayed next to the login form, which could be considered misleading – impersonating a secure connection). Passwords are stored as MD5 hashes with 33 characters long alphanumeric hash, random and unique for each user. The author did not attempt to find a plugin preventing session hijacking – SSL support was already included, and thus complete (compare with Drupal, where SSL had to be manually added).

Figure 3.4: VirtueMart – URL rewriting filter

Source: Author, VirtueMart .htaccess file.

The downloaded bundle\(^{21}\) (and an original, unbundled package\(^{22}\)) both contained a three months old SQL injection vulnerability\(^{23}\) – a patch\(^{24}\) has been released, but only as a few files and a regular announcement on the page (as of April, hidden bellow several other announcements). Original vulnerable files are still marked as the official release. Patches (see diff output in Figure 3.5, Figure 3.6) did not contain any change logs nor were included in the version history of VirtueMart.

Exceptions are suppressed in file manipulation functions (VirtueMart source)\(^{25}\) and comments could be improved (in some files, only public interfaces were described). Some of the plugins in Joomla attempted to execute other programs (\texttt{wget}, list of running programs, etc.) but not the shopping cart itself, so it is not penalized in the evaluation. CAPTCHA is available as a separate plugin.

Multiple “security” extensions are available, at least three of them only adding additional password for admin interface (which might imply a weak security track). CSRF is not consistent (protection is available, but VirtueMart sources contain forms where CSRF tokens are not added). SQL sanitation is handled manually at the top level, not within the database abstraction. MIME headers are not checked

---


\(^{22}\)http://dev.virtuemart.net/attachments/download/18/VirtueMart_1.1.4-COMPLETE_PACKAGE.j15.zip (last retrieved April 29, 2010).


\(^{25}\)E.g. administrator/components/com_virtuemart/Tar.php.
Figure 3.5: VirtueMart – SQL injection I (diff output for patch and current version)

```php
require_once(CLASSPATH . 'ps_order_status.php');
$ps_order_status = new ps_order_status();
$order_status_id = vmGet($REQUEST, 'order_status_id ');
+$order_status_id = vmrequest::getInt('order_status_id', 0);
$option = empty($option)?vmGet($REQUEST, 'option', 'com_virtuemart')?:$option;
// First create the object and let it print a form heading
```


Figure 3.6: VirtueMart – SQL injection II (diff output for patch and current version)

```php
$itemid = $sess->getShopItemid();
$db_product = new ps_DB;

+ // Check for non-numeric product id
+ if (!empty($product_id)) {
+   + if (!is_numeric($product_id)) {
+     $product_id = '';
+   } +
+}
+ // Get the product info from the database
+ $q = "SELECT * FROM '#_vm_product' WHERE ";
+ if (!empty($product_id)) {
+   $product_name = substr($product_name, 0, 64);
+   
+   $vm_mainframe->setPageTitle(html_entity_decode($page_title, ENT_QUOTES, vmGetCharset()));
+   +$vm_mainframe->setPageTitle(@html_entity_decode($page_title, ENT_QUOTES, vmGetCharset()));
+   // Prepend Product Short Description Meta Tag "description"
+   if (vmIsJoomla('1.5')) {
```

nor enforced consistently (there are several upload modules, each handling files in its different way – VirtueMart has a separate image upload module!). Multiple data leaks were detected by the author, but would not be visible had the error reporting been disabled as recommended in the setup.

Joomla does not provide granular permissions (only nine predefined groups and no per action rights, which is completely insufficient for an e-commerce package).

VirtueMart at its current state contains a security vulnerability acknowledged by its authors and so its use for e-commerce is discouraged. The whole package seemed to the author to be very loosely interconnected, and so it may contain many more bugs, as most of the code is duplicated at multiple places.

3.3.7 Zen Cart

Zen Cart, based on osCommerce\(^{26}\) (forked in 2003\(^{27}\)), is one of the oldest (by age, code and “design” style) shopping carts in the assessment. In size and functions, it is comparable to Oxid eshop (e.g. less user-oriented and more focused on merchants).

Installation manual is not kept up to date (instructions on writeable directories and configuration files are different in the installation script). SSL can be configured during installation (even though the installation itself is over unencrypted channel), which slightly increases security (prevents the first unencrypted administrator’s login). Renaming administrator’s interface folder is recommended.

Access to several world-writeable directories is restricted through .htaccess directives (Figure 3.7), which duplicates protection in uploading scripts and does not prevent MIME sniffing vulnerability (uploading functions compare user-supplied MIME type with file extension, not the actual content).

Figure 3.7: Zen Cart, .htaccess

```
<FilesMatch ".*">
  Order Allow,Deny
  Deny from all
</FilesMatch>

# but now allow just *certain* necessary files:
<FilesMatch ".*\(jpe?g|JPE?G|gif|GIF|png|PNG|swf|SWF\)$” >
  Order Allow,Deny
  Allow from all
</FilesMatch>
```

Source: Author.

Session tokens are protected against eavesdropping (see Appendix E for session

\(^{26}\)http://www.oscommerce.com/about/about (last retrieved April 29, 2010).
regeneration procedure), however, the newly regenerated token could be susceptible to a weakness in PHP `rand()` function on certain Windows versions\(^{28}\).

Installation is logged in detail, but no other logging is provided. Granular permissions are non-existent – a plugin is available, but only compatible with Zen Cart up to 1.3.8. Passwords are stored as MD5 hashes with 2 characters lower-case hash, which might not be completely secure (`zen_rand()` returns a random number < 100000). Similar schemes were found in previous e-commerce software packages, but this hash generation seems to be particularly weak.

![Figure 3.8: Zen Cart – password hashes](source)

```php
function zen_encrypt_password($plain) {
    $password = '';  
    for ($i=0; $i<10; $i++) {
        $password .= zen_rand();
    }
    $salt = substr(md5($password), 0, 2);
    $password = md5($salt . $plain) . ':' . $salt;
    return $password;
}
```

*Source: Author, Zen Cart 1.3.9b, admin/includes/functions/password_funcs.php, l. 29+

Exceptions are suppressed (especially file manipulation). There are attempts in the code to directly execute `ImageMagick` and `sendmail` and avoid built-in PHP functions. Database queries are escaped inconsistently by `addslashes()`, which is not sufficient in case of multi byte characters (compare with `mysql_real_escape_string()`). If wrong administrator password is supplied, the system adds delay before the form can be submitted again. However, for correct password, no delay is present, so an attacker can still guess passwords faster by considering a certain delay as an unsuccessful attempt. HTML filtering is inconsistent (see Figure 3.9).

![Figure 3.9: Zen Cart – HTML escaping](source)

```html
<tr><td class="smallText"><b><?php echo TEXT_FROM; ?></b><br/><?php echo htmlspecialchars(stripslashes($_POST['from'])); ?></td></tr>
```

*Source: Author, Zen Cart 1.3.9b, example.*

New user registration requires age by default – such requirement is completely

\(^{28}\) The author has verified that PHP `rand()` generates a regular pattern (if random numbers are used to plot points in 2D space) for Windows XP, Windows 2003 and Windows 2008, which can be interpreted as a cryptographic weakness; `mt_rand()` does not produce any (visible) patterns.
unnecessary and increases the burden of protecting private data. No CAPTCHA plugin to prevent spam exists for current version of Zen Cart. Patch distribution is questionable, as has been mentioned earlier at the beginning of Chapter 3. No security mailing list is available.

Overall, Zen Cart “is showing its age” and most of the problems present in other shopping carts separately have accumulated. The author would not recommend using Zen Cart for production environment and legacy systems should be migrated to a more secure software.

3.4 Remarks

Software in the assessment varied widely, but certain patterns can be distinguished. The clear winner was Ubercart thanks to Drupal core, but near-winners were Magento and Oxid e-shop. The most important problems encountered were ineffective input sanitation (mainly HTML), which can be very easily mitigated by using a templating engine and automatically escaping any input unless specified differently by the programmer. The other approach (selectively sanitizing) is prone to errors. Secure transmission of session tokens had been accomplished only in two cases (Ubercart, Zen Cart), despite the fact that implementing it directly involves altering only a single part of the application (authentication). The author believes that session hijacking is not considered to be important by application writers as it cannot be exploited automatically. However, with the advent of wireless networks, this technique could become more widespread.

Lack of documentation and security handling policy proved to be common and can also be improved with very small costs – application users should always know the security risks involved. All software but Magento and Zen Cart did not respect the \[10\] directive (unsolicited email and informed consent). The code of all applications was reasonably readable, but some of them violated “best practices” (see \[32\]; mainly suppressing exceptions). Use of outdated cryptographic functions (MD5) with dubious salts is left up to further discussion – the author does not possess the knowledge and was not able to find any hints in literature as to whether “breaking” MD5 with two character salt is computationally feasible or not (it is secure against stock rainbow tables attack, but with simple passwords and determined attacker, it is debatable as to whether the protection would stand)\cite{29}.

\footnote{The author believes that for most e-commerce applications, a “brute-force” dictionary attack with a list of most common passwords (see \cite{24} for many examples) would be feasible even with 2 character random alphanumeric salt (the lookup speed would be very similar to plain dictionary attack, the size of the dictionary still remains in order of gigabytes and generating of dictionaries can be run in parallel e.g. over a botnet).}
Considerably fewer security vulnerabilities were found if the software built upon an existing frameworks (Ubercart – Drupal, Magento – Smarty) than if it created everything from scratch (Zen Cart). A comparison of “best” package with “average” is provided in Figure 3.10

**Figure 3.10:** Comparison of best and average software package

For production use, the current writer recommends (despite the flaws) Magento, Oxid e-shop or Ubercart, depending on the website role (Ubercart is well suited for content-rich sites, Magento or Oxid e-shop for classic e-commerce with large stock). Other packages should be avoided in production environment and legacy sites should be moved.

It has been found by the author that all packages attempted to achieve some security, but the actual level varied. With some packages, site owners would be vulnerable to automatic mass-scale attacks that could endanger the actual business based on the site; with others (three recommended packages above), the level of risk could be established as “acceptable” for the type of application, size of the business and financial investment. It is arguable whether the risk may be transferable to the application service provider through a contract, but a full coverage is unlikely and the residual risk is present for most small e-commerce sites and corresponds to the stakes.
Chapter 4

User perception of security

To establish the perception of computer security of actual users and accomplish the goal of the thesis by evaluating whether security affects turnover from the users’ point of view, a small survey has been conducted by the author. It should be considered an exploratory study, rather than an empirical research because of the size of the poll group, number of questions and quantifiability (a similar, older study with many more participants is in [35]). However, the current writer still believes that it captures important trends and allows to make moderate conclusions.

A type of online survey, filled in the browser, was selected in order to allow as many replies as possible and only twelve questions were asked in order to not make the survey time-consuming (despite that, about 50 % of visitors of the survey did not respond and another 20 % did not finish the survey).

Six general topics have been covered:

- information about the respondent
- message confidentiality and integrity
- malicious code
- private data
- e-commerce self-certification
- user tracking
- passwords.

The full survey is available on the enclosed DVD. A total of 77 valid responses has been analysed, another 41 partial and 3 untrustworthy (always the first option) responses were discarded. If any screenshots were presented to the user, Internet Explorer 8 or Firefox 3 running on a Czech version of Windows XP with default themes were employed (most common combination, based on statistics) – the goal was to not confuse users. For sites, Facebook was preferred as a platform, because most
respondents can be expected either to own Facebook account or to have observed somebody use Facebook and thus should be familiar with the site.

Further research regarding user trust can be found in [14] (effects of presentation flaws).

The questions will now be described in detail, together with results.

4.1 Survey questions and results

Information about the respondent attempted to divide respondents into different groups based on age range (0-18, 18-25, 26-35, . . . , 56-65, 65+), self-assessed computer knowledge (5 levels: no knowledge, limited use – self-taught, basic use with some computer courses during lifetime, power user, work or education experience in the IS/ICT field) and self-assessed amount of computer usage for personal purposes (1 hour per week, 2-5 hours per week, 1 hour daily, 2 hours daily, 2+ hours daily). The study was mostly conducted among young university students (while attempting to avoid IS/ICT students), so the results are valid for people with general and “normal” computer skills, with at least limited amount of computer courses at high school or university (mainly office information software) and with daily exposure to computers. For “computer-challenged” users, the results should be expected to be worse in terms of security.

Table 4.1: Self-assessed computer skills level

<table>
<thead>
<tr>
<th>Skill level</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>No knowledge of computers</td>
<td>2</td>
</tr>
<tr>
<td>Limited use, self-taught</td>
<td>10</td>
</tr>
<tr>
<td>Basic use with some computer courses during lifetime</td>
<td>22</td>
</tr>
<tr>
<td>Power user</td>
<td>36</td>
</tr>
<tr>
<td>Work or education experience in the IS/ICT field</td>
<td>7</td>
</tr>
</tbody>
</table>

Table 4.2: Age ranges

<table>
<thead>
<tr>
<th>Age range</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Less than 18 years</td>
<td>5</td>
</tr>
<tr>
<td>18 - 25</td>
<td>58</td>
</tr>
<tr>
<td>26-35</td>
<td>5</td>
</tr>
<tr>
<td>36-45</td>
<td>4</td>
</tr>
<tr>
<td>46-55</td>
<td>3</td>
</tr>
<tr>
<td>56-65</td>
<td>2</td>
</tr>
<tr>
<td>66+</td>
<td>0</td>
</tr>
</tbody>
</table>
4. User perception of security

Table 4.3: Time spent using computers for personal purposes

<table>
<thead>
<tr>
<th>Amount of time</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Less than 1 hour per week</td>
<td>0</td>
</tr>
<tr>
<td>2 - 5 hours per week</td>
<td>3</td>
</tr>
<tr>
<td>1 hour daily</td>
<td>3</td>
</tr>
<tr>
<td>2 hours per day</td>
<td>11</td>
</tr>
<tr>
<td>More than 2 hours per day</td>
<td>60</td>
</tr>
</tbody>
</table>

Message confidentiality and integrity included two tests – the first one (question 4) presented users with a choice of two Facebook login pages in Internet Explorer 8, one loaded over regular connection (Figure G.1) and one over SSL encrypted connection (Figure G.2). The respondents were asked to choose which option probably\(^1\) prevents third parties from eavesdropping (aimed at the recognition of SSL).

Two (unsurprising) trends are present in the data – with growing computer skill, users become more confident in recognizing “safer” (encrypted) pages, albeit it is up to discussion whether they actually grasp the concept. A total of 34% (and both “unskilled” users) considered both pages to be insecure and a further study should be conducted as to why this happens (the author believes it is a psychological problem of posing questions about security, as people tend to think more about vulnerabilities when asked to).

The second question (number 5) presented a self-signed certificate warning (Figure G.3) in Firefox 3\(^2\) and asked the respondents to describe in their own words what the warning means. The warning was allegedly displayed after attempting to access web page of The University of Economics in Prague (site chosen to increase confidence).

Overall, three categories of responses can be distinguished: “no understanding” (responses suggesting leaving the site, security problems, empty responses, roughly 50%), “blind acceptance” (recommending adding an exception and trusting the certificate, blaming the browser, roughly 45%) and at least “partial understanding” (responses mentioning self-signed certificate or possible man-in-the-middle attack, about 5%). There was not a clear distinction between “unskilled” and “self-taught” groups (no clear answer) and “skilled” and “power-users” (accepting certificate, leaving the site, no clear answer). Alarmingly, in the “IS/ICT” group only three out of seven respondents answered correctly.

\(^1\) The word “probably” was specifically included as users never know whether the information would be submitted over encrypted channel (newer browsers emit a warning if not, but not before submitting the form).

\(^2\) Internet Explorer 8 displays an error identical to malfunctioning network connection, hence it is unusable for the test.)
The conclusion is that many users probably cannot recognize a secure site during casual browsing, and that presenting a self-signed certificate would rapidly decrease conversion rate (compared to no encryption at all). The number of users with “security doubts” might indicate that increasing the amount of “assurances” on the page and advertising high security could help casual buyers to choose the selected site for shopping (see passwords and self-certification; results are consistent with previous research e.g. in [38]).

Table 4.4: SSL recognition in relation to self-assessed computer skills (question number 4)

<table>
<thead>
<tr>
<th>Skill and choice</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>No knowledge of computers</td>
<td>2</td>
</tr>
<tr>
<td>Both pages are equally insecure</td>
<td>2</td>
</tr>
<tr>
<td>Limited use, self-taught</td>
<td>8</td>
</tr>
<tr>
<td>Both pages are equally secure</td>
<td>1</td>
</tr>
<tr>
<td>Both pages are equally insecure</td>
<td>4</td>
</tr>
<tr>
<td>The lower page is more secure</td>
<td>5</td>
</tr>
<tr>
<td>Basic use with some computer courses during lifetime</td>
<td>16</td>
</tr>
<tr>
<td>Both pages are equally secure</td>
<td>3</td>
</tr>
<tr>
<td>Both pages are equally insecure</td>
<td>6</td>
</tr>
<tr>
<td>The upper page is more secure</td>
<td>1</td>
</tr>
<tr>
<td>The lower page is more secure</td>
<td>11</td>
</tr>
<tr>
<td>Power user</td>
<td>31</td>
</tr>
<tr>
<td>Both pages are equally secure</td>
<td>4</td>
</tr>
<tr>
<td>Both pages are equally insecure</td>
<td>9</td>
</tr>
<tr>
<td>The upper page is more secure</td>
<td>1</td>
</tr>
<tr>
<td>The lower page is more secure</td>
<td>22</td>
</tr>
<tr>
<td>Work or education experience in the IS/ICT field</td>
<td>7</td>
</tr>
<tr>
<td>Both pages are equally insecure</td>
<td>1</td>
</tr>
<tr>
<td>The lower page is more secure</td>
<td>6</td>
</tr>
</tbody>
</table>

Private data studied the users’ perception of handing out personal details in a “hostile” environment (e-commerce site) and in a friendly, well known place over a period of time (social networking site). Respondents were presented first with a form asking what information they would submit (without fear) to an e-commerce site. Later on in the questionnaire (at the next page), they were to estimate what information could be collected about them on social networking sites. Possible answers were name, address, telephone, email, date of birth, a Czech equivalent of social security number, number of their ID card, payment card number, bank
account number, nationality and ethnic origin, political opinions, union membership, religion, health and information about sexual life (options selected from [29] and [11]).

To quantify the data, boolean values (0 – user would not / does not provide data, 1 – would provide / provides data) were assigned to possible answers, values were summed up and an average was calculated to compare where users supply more data. With higher computer skills, respondents (power users) tended to trust e-commerce vendors (4.08) and to trust the least the social networking sites (2.67). With lower computer skills, much more (3.4 – self-taught, 3.2 – basic usage) data was supplied to e.g. Facebook. IS/ICT groups and “unskilled” are statistically unverifiable.

As to actual private data, 23 % respondents would not provide address and 46.8 % telephone to the online store. The first value is probably invalid in a real world situation, but the second decreases the possibility of validating orders by telephone. E-commerce owners should, again, increase the trust of customers or offer an additional benefit for providing the number (e.g. order confirmation). Also, another implication from the data is that if age group is required for the business purposes, it can be collected from social networking sites (where roughly 45.5 % of respondents provide it, compared to 24.4 % in case of e-commerce sites). Most users seemed to provide a very limited amount of data to social networking sites, but the real amount is unknown.

The whole data set is stored on the enclosed CD and not included here due to size concerns (readability).

**Malicious code** displayed a Facebook login page (Figure G.4) with a “leaked” JavaScript code at the bottom (actually a part of Troj/JSRedir-R (Gumblar)³), suggesting the page has been altered, to test whether respondents recognize differences in a common page (a sign of a possible unwanted modification). There were four possible combinations of “real / fake Facebook login page” and “can / cannot damage my computer” as answers.

All power users recognized the code and identified the page as either malicious or modified. With decreasing computer skills, the amount of users mistakenly considering the page to be secure increased (to 16 % in case of power users, 26 % and 57 % in case of self-taught and “no knowledge” groups). It is a matter of further research if respondents actually recognized the code or replied according to what they thought they were expected to reply.

The conclusion for e-commerce sites is unknown – it is not clear whether users would attribute the code to a security problem or a bug (however, by itself the

³Recovered by the author from an infected site.
presence of such malicious code will trigger a removal from search engines, e.g. Google, when discovered by the crawler and remains dangerous).

**Self-certification** was chosen as a sample of “legal requirements”, both because it is a recognisable feature and because, at least in the Czech Republic, it mostly comprises of a verification of already established legal requirements. Also, it would be next to impossible to test the actual role of recognition of consumer rights in such a small scale study. The question has been inspired by work about web assurance seals in [21] (albeit the authors of the study approach the problem from a different angle).

Several logos were selected (APEK⁴, SOS⁵ as well-known brands, and four others at random) and respondents were asked whether a logo on the page increases their trust and confidence. Respondents could choose from four levels of “trust” (“well known logos increase trust”, “any logo increases trust”, “no trust increase because a logo does not guarantee anything” and “no trust increase because any certification can be bought”).

There was no identifiable distinction between “skill groups” this time. Only 37 % of respondents trust the logos, which might be attributed to the formulation of the question rather than consumer trust and may be quite different in real world situations (especially for well-known certifications, or certifications with a lot of promotion).

**Tracking** checked if users remove (and can remove) any cookies or LSO, used for tracking. Respondents were asked what data they regularly delete from their browser (with options “none”, “history”, “cache”, “cookies”, “LSO cookies”).

<table>
<thead>
<tr>
<th>Type of data</th>
<th>Share</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nothing</td>
<td>29.8 %</td>
</tr>
<tr>
<td>History</td>
<td>50.6 %</td>
</tr>
<tr>
<td>Cookies</td>
<td>50.6 %</td>
</tr>
<tr>
<td>LSO (flash) cookies</td>
<td>31.2 %</td>
</tr>
<tr>
<td>Cache</td>
<td>39.0 %</td>
</tr>
</tbody>
</table>

To verify claims about deleting LSO, ten randomly selected users from the IS/ICT or power user skill group, who provided email address, were contacted by


⁵Certification from a consumer rights’ group.
the author and asked to check their computer for presence of LSO cookies. Only two users’ computers were devoid of LSO, one due to manual cleaning and one thanks to CCleaner program. Three power users were not able to locate the directory with LSO objects at all (probably due to hidden files).

The conclusion is that cookies are becoming less dependable for tracking knowledgeable users (again, the author would like to stress that the study contained mainly knowledgeable computer users), as many of them clear browser temporary files from time to time, which lowers the credibility of affiliate programs. This also creates another problem: assigning user accounts to returning customers and storing shopping carts. LSO’s can replace cookies only as an auxiliary measure and bring their own risks – e-commerce site owners should very carefully track the development of laws and the legal system, because LSO’s cannot be easily deleted and provide a large, and probably still undiscovered, target for establishing privacy rights (regular cookies being already covered).

**Passwords** section inquired respondents about their (self-assessed) password complexity and use. Five levels were possible (“one password for all uses”, “one password for all services with some exceptions”, “one strong password for important and one simple for other services”, “multiple strong password for important and simple for other services”, “always a unique password for every service”). The strength of the password is questionable (and cannot be ethically verified). Only one respondent uses a single password. One password is for most services used by 25% and for all important services by 26% of users. In reality, the amount could be higher, because customers may subconsciously add more importance to the e-commerce user account than it actually bears.

E-commerce site owners are at risk of storing potentially sensitive data, and with careless customers, the implications of losing customer database with poorly stored passwords (see hashes in Chapter 2) would be grave.

### 4.2 Remarks

It has been verified in a limited study that security can affect turnover (either by increasing customer trust or by unnecessarily scaring potential customers). Two distinctive possibilities for improvement and increasing the turnover exist: maintaining “invisible” security (e.g. correct password storage or verified source code of the web page) and selling “visible” (perception of) security as a competitive advantage by convincing the users (see [38]). If the online vendor invested into certification, the user should be provided with a way to verify the claim, or else the certification loses
4. User perception of security

credibility. E-commerce sites should also maintain validity and be well tested in all current browsers, especially in “security-critical” parts.6

6By anecdotal evidence, maintaining basic security is problematic even for large e-commerce companies. To name a few: alza.cz (turnover of over 4 billion CZK in 2009) – SSL encrypted login, but with browser errors (parts of the page are not encrypted); mall.cz (turnover of over 1.5 billion CZK in 2009) – no SSL login at all; source of turnover data: http://www.lidovky.cz/vanoce-miliardy-pro-e-shopy-dfr-/ln_noviny.asp?c=A100118_000063_ln_noviny_sko (last retrieved May 25, 2010).
Chapter 5

Conclusion

The goals of the thesis have been generally achieved by the author. “Acquis communautaire” legislation of the European Union related e-commerce security (and privacy) has been laid out and common security issues of web applications were explored in sufficient detail to provide a theoretical background for the practical part of the thesis.

The first method, proposed by the author to study security of small e-commerce security sites, was only partially successful. The current writer has demonstrated that the use of honey-pot on a small scale did not bring comprehensive results, as the exploitability of vulnerabilities strongly depends on (innumerable) server configurations and with a well-configured server, common attacks were not effective and hence did not provide enough data. An alternative method, evaluation of common open-source packages, based on six topics (each with four subcriteria, see Section 3.1), did provide enough findings to justify making a conclusion about security despite the small coverage of the criteria (especially source code analysis).

It has been found out by the author that security of open-source small e-commerce shopping carts (packages) varies wildly and does not correspond to the install base. The software with the worst score (OpenCart, Prestashop, Zen Cart, VirtueMart) did not accomplish basic security criteria and in one case (VirtueMart), a version with a publicly known vulnerability had been distributed for three months (without a patch; as of April 2010). A reasonable level of security is provided by three packages (UberCart, Oxid e-shop, Magento), but even for these, a higher score could have been achieved even with a very limited endeavor. The most common issues were documentation, input sanitation CSRF and lagging behind current cryptographic trends.

The effect of security on turnover has been studied by means of an exploratory study. Despite a limited set of participants (the results include mostly computersavy users), clear trends were identified by the author (general dependance on sym-
bols of security rather than deep understanding, ie. perceived security). The results provide a helpful starting point for a more comprehensive study.

For further research, the author would like to recommend two distinctive areas. With enough resources, a honey-pot survey might provide sufficient results for analysis. However, more data could be collected by creating a web crawler to check common (e.g. top 50 for a set of keywords) e-commerce sites for signs of security breaches (an open study is typically not possible, either because of closed-source applications or reliability and temporariness of disclosed “security leaks”).

As for the turnover study, the author proposes a study in a closed, controlled environment (due to the limits of surveys) in a form of game, where participants spend a preset amount of money by shopping at several sites, each with similar products but different design and security properties. Such study will provide invaluable data if carefully balanced (or if executed in several rounds).
Bibliography


[15] FURNELL, S. M. The Problem of Categorising Cybercrime and Cybercrimi-


Appendix A

SSL performance impact

A test of difference between encrypted and non-encrypted communication has been carried out by the author. First, a test file was created (roughly 10 kB) with random data (to prevent any compression). The size of the file was chosen as it would stress the establishment of SSL connection rather than the actual symmetric encryption (larger files). Then, ApacheBench (2.3) was executed with 200 threads and 10 000 requests in total from one “client” to a server (C2D dual-core 2.33 GHz, 2 GB RAM, normal load average: 0.02, 0.01, 0.01) over OVH network (100 mbps connection, with not more than 10 % used at the time). Encrypted connection was five times slower, with load reaching 30 on the server. The test should be considered just as an example, not an empirical study; despite the considerable inaccuracy, it still proves that there is significant performance difference between encrypted and unencrypted connections.

Figure A.1: File – 10240 Bytes

dd if=/dev/urandom of=file bs=1K count=10

Source: Author.

1http://www.ovh.com
A. SSL performance impact

Figure A.2: Encrypted vs non-encrypted communication

ab -n 10000 -c 200 http://www.domain.cz/file

Time taken for tests: 9.340 seconds
Failed requests: 0
Requests per second: 1070.63 [#/sec] (mean)
Time per request: 186.806 [ms] (mean)
Time per request: 0.934 [ms] (mean, across all concurrent requests)
Transfer rate: 10994.84 [Kbytes/sec] received

Connection Times (ms)

<table>
<thead>
<tr>
<th></th>
<th>min</th>
<th>mean [+/-sd]</th>
<th>median</th>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connect</td>
<td>1</td>
<td>37.0</td>
<td>36.0</td>
<td>3038</td>
</tr>
<tr>
<td>Processing</td>
<td>5</td>
<td>138.0</td>
<td>145.0</td>
<td>405</td>
</tr>
<tr>
<td>Waiting</td>
<td>3</td>
<td>100.0</td>
<td>103.0</td>
<td>378</td>
</tr>
<tr>
<td>Total</td>
<td>8</td>
<td>175.0</td>
<td>183.0</td>
<td>3184</td>
</tr>
</tbody>
</table>

ab -n 10000 -c 200 https://www.domain.cz/file

SSL/TLS Protocol: TLSv1/SSLv3, DHE-RSA-AES256-SHA, 1024, 256

Time taken for tests: 45.590 seconds
Failed requests: 0
Requests per second: 219.34 [#/sec] (mean)
Time per request: 911.809 [ms] (mean)
Time per request: 4.559 [ms] (mean, across all concurrent requests)
Transfer rate: 2251.66 [Kbytes/sec] received

Connection Times (ms)

<table>
<thead>
<tr>
<th></th>
<th>min</th>
<th>mean [+/-sd]</th>
<th>median</th>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connect</td>
<td>25</td>
<td>660.0</td>
<td>590.0</td>
<td>4593</td>
</tr>
<tr>
<td>Processing</td>
<td>2</td>
<td>248.0</td>
<td>250.0</td>
<td>1954</td>
</tr>
<tr>
<td>Waiting</td>
<td>1</td>
<td>103.0</td>
<td>80.0</td>
<td>1954</td>
</tr>
<tr>
<td>Total</td>
<td>67</td>
<td>907.0</td>
<td>870.0</td>
<td>4747</td>
</tr>
</tbody>
</table>

Source: Author; Apache/2.2.9 (Debian).
Appendix B

Eavesdropping – traffic statistics

The following traffic statistics shows that malfunctioning network equipment can be easily detected if changes of runtime data are regularly checked. Note: with 10 mbps connection, the traffic will cause considerable lagging with even simple tasks (e.g. basic SSH usage).

Figure B.1: Networking equipment malfunction – traffic

<table>
<thead>
<tr>
<th>day</th>
<th>rx</th>
<th>tx</th>
<th>total</th>
</tr>
</thead>
</table>
| 15.10. | 360.85 MB | 2.89 GB | 3.25 GB :
| 16.10. | 413.79 MB | 2.83 GB | 3.23 GB :
| 17.10. | 353.40 MB | 2.58 GB | 2.92 GB :
| 18.10. | 370.34 MB | 3.44 GB | 3.80 GB :
| 19.10. | 527.91 MB | 4.17 GB | 4.69 GB :
| 20.10. | 360.02 MB | 2.82 GB | 3.17 GB :
| 21.10. | 396.22 MB | 3.89 GB | 4.27 GB :
| 22.10. | 406.98 MB | 2.94 GB | 3.34 GB :
| 23.10. | 368.23 MB | 3.65 GB | 4.01 GB :
| 24.10. | 404.95 MB | 3.25 GB | 3.64 GB :
| 25.10. | 381.37 MB | 3.45 GB | 3.83 GB :
| 26.10. | 427.78 MB | 5.54 GB | 5.96 GB :
| 27.10. | 462.06 MB | 5.54 GB | 5.99 GB :
| 28.10. | 587.34 MB | 6.86 GB | 7.44 GB :
| 29.10. | 1.18 GB  | 7.54 GB | 8.72 GB %::
| 30.10. | 13.30 GB | 6.10 GB | 19.40 GB XXXXXX:
| 31.10. | 9.45 GB  | 6.35 GB | 15.79 GB XXXXX:
| 01.11. | 11.64 GB | 7.55 GB | 19.19 GB XXXXX:
| 02.11. | 13.72 GB | 10.60 GB | 24.32 GB XXXXXXXX:%:
| 03.11. | 7.73 GB  | 8.94 GB | 16.67 GB XXXXX:
| 04.11. | 11.63 GB | 9.51 GB | 21.14 GB XXXXX:
| 05.11. | 12.23 GB | 8.41 GB | 20.64 GB XXXXX:
| 06.11. | 10.46 GB | 8.23 GB | 18.69 GB XXXXX:
| 07.11. | 16.90 GB | 8.09 GB | 24.99 GB XXXXXXXX:%:
| 08.11. | 42.85 GB | 7.79 GB | 50.65 GB XXXXXXXXXXXXXXXXXXXX::%
| 09.11. | 24.26 GB | 2.72 GB | 26.98 GB XXXXXXXXXXXX:

---

**estimated** 46.77 GB | 5.24 GB | 52.01 GB

*Source:* Author; vnstat, Server4u.cz malfunctioning equipment.
Appendix C

PAP intercepted communication

POST /scripts/server.php HTTP/1.1
Host: members.qualityunit.com
Accept: */*
Content-Length: 203
Content-Type: application/x-www-form-urlencoded

D="("C":"GpfRpcServer","M":"run","requests":{""C":"Dp_QualityUnit_AddonPurchase","M":"getAddonsList","productId":"ffd43d92"})"

HTTP/1.1 200 OK
Date: Tue, 27 Apr 2010 14:00:53 GMT
Server: Apache/2.2.11 (Unix) mod_ssl/2.2.11 OpenSSL/0.9.7a mod_bwlimited/1.4 PHP/5.2.9
X-Powered-By: PHP/5.2.9
Set-Cookie: _dp_sid=87ca831d3ed8979edfab08b287bbcca; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Content-Length: 43
Content-Type: text/html

HTTP/1.1 200 OK
Date: Tue, 27 Apr 2010 14:00:54 GMT
Server: Apache/2.2.11 (Unix) mod_ssl/2.2.11 OpenSSL/0.9.7a mod_bwlimited/1.4 PHP/5.2.9
X-Powered-By: PHP/5.2.9
Set-Cookie: _dp_sid=b6be6bd86662d5d5c9e0dd5f0a1422d8; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Content-Length: 334
Content-Type: text/html

[[["name","value"],["product_name","Post Affiliate Pro"],["app_code","pap"],["variation_name","Xpress edition (Owned)"]],["license","Gu7WSwMDAyVMdzMg4YD/ki7BhMwтроQnilS4bsIydDSFxA1Wu++wrK8+KF6682b380nDFWBMbV+]]
C. PAP intercepted communication

```
POST /scripts/server.php HTTP/1.1
Host: members.qualityunit.com
Accept: */*
Content-Length: 246
Content-Type: application/x-www-form-urlencoded

D={"C":"Gpf_Rpc_Server","M":"run","requests":[
  {"C":"Dp_Version_Info","M":"getLatestVersion","data":[]}
]}
```

```
POST /scripts/server.php HTTP/1.1
Host: members.qualityunit.com
Accept: */*
Content-Length: 346
Content-Type: application/x-www-form-urlencoded

D={"C":"Gpf_Rpc_Server","M":"run","requests":[
  {"C":"Dp_License_Generator","M":"validateLicense","data":[]}
]}
```
Appendix D

Test server configuration

Table D.1: Relevant test server configuration

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Intel i7 920, 8 cores, 2.66 GHz, 8 GB DDR3 RAM (no ECC), 1 Gbps network connection (ovh.net)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating system</td>
<td>Debian GNU/Linux 5.0.4 (Lenny), x86_64</td>
</tr>
<tr>
<td>Kernel</td>
<td>2.6.3.11 (vanilla) with grsec patches applied.</td>
</tr>
<tr>
<td>Configuration notes</td>
<td>The server is using both IPv4 and IPv6. No specific cryptographic hardware (accelerator) was installed.</td>
</tr>
<tr>
<td>Webserver</td>
<td>Apache 2, 2.2.9-10+lenny7</td>
</tr>
<tr>
<td>Webserver – modules</td>
<td>libapache2-mod-gnutls (0.5.1-1), libapache2-mod-antiloris (0.4-1), libapache2-mod-evasive (1.10.1-1)</td>
</tr>
<tr>
<td>Database</td>
<td>MySQL, 5.0.51a-24+lenny3</td>
</tr>
<tr>
<td>PHP</td>
<td>PHP5, 5.2.6.dfsg.1-1+lenny8, using libapache2-mod-php5 (5.2.6.dfsg.1-1+lenny8) to interface with Apache</td>
</tr>
<tr>
<td>PHP – patches</td>
<td>Suhosin, 0.9.27-1</td>
</tr>
<tr>
<td>PHP – op-cache</td>
<td>Xcache, 1.2.2-3</td>
</tr>
</tbody>
</table>

Source: Author; only relevant software mentioned; full configuration is supplied on the accompanying CD.

PHP is configured according to recommendations at http://php.net/manual/en/security.php, other changes (stricter rules) have been performed by the author and are based on PHP manuals.
Figure D.1: PHP configuration

```
short_open_tag = On  
asp_tags = Off       
precision = 12       
output_buffering = Off  
zlib.output.compression = Off  
zlib.output.compression_level = 0  
imPLICIT_flush = Off  
allow_call_time_pass_reference = On  
safe_mode = Off  
safe_mode_gid = Off  
safe_mode_include_dir =  
safe_mode_exec_dir =  
safe_mode_allowed_env_vars = PHP  
safe_mode_protected_env_vars = LD_LIBRARY_PATH  
disable_functions = (same as suhosin.ini)  
disable_classes =  
expose_php = Off  
max_execution_time = 30  
max_input_time = 60  
memory_limit = 64M  
error_reporting = E_ALL & ~E_NOTICE  
display_errors = On  
register_globals = Off  
register_long_arrays = Off  
register_argc_argv = Off  
post_max_size = 8M  
magic_quotes_gpc = On  
magic_quotes_runtime = Off  
magic_quotes_sybase = Off  
enable_dl = Off  
file_uploads = On  
upload_max_filesize = 16M  
allow_url_fopen = Off  
allow_url_include = Off  
```

code/php.ini

Source: Author, only relevant configuration options shown.
Figure D.2: PHP configuration, cont.

```ini
[Session]
session.save_handler = files 1
session.use_cookies = 1 2
session.name = PHPSESSID 3
session.auto_start = 0 4
session.cookie_lifetime = 0 5
session.cookie_path = / 6
session.cookie_domain = 7
session.cookie httponly = 8
session.serialize_handler = php 9
session.gc_divisor = 100 10
session.gc_maxlifetime = 1440 11
session.bug_compat_42 = 0 12
session.bug_compat_warn = 0 13
sessionreferer.check = 14
session.entropy_length = 0 15
session.entropy_file = 16
session.cache_limiter = nocache 17
session.cache_expire = 180 18
session.use_trans_sid = 0 19
session.hash_function = 0 20
session.hash_bits_per_character = 4 21
urlrewriter.tags = "a=href,area=href,frame=src,input=src,form=,fieldset=" 22
```

code/php.sess.ini

Source: Author, only relevant configuration options shown.
Figure D.3: Suhosin configuration

```ini
[ suhosin ]
suhosin.log.syslog = 511
suhosin.executor.func.blacklist = exec, apache.get_modules, apache.get_version, apache.getenv, apache_child_terminate, apache_note, apache_setenv, closelog, define_syslog_variables, dl, escapeshellcmd, escapeshellarg, exec, ini_alter, leak, link, openlog, passthru, pclose, pcntl_exec, pfsockopen, popen, posix_getpwnam, posix_getpwuid, posix_getgrnam, posix_kill, posix_mkfifo, posix_mknod, posix_times, proc_close, proc_get_status, proc_nice, proc_open, proc_terminate, shell_exec, symlink, syslog, system, virtual, readlink, php_uname, getmyuid, getmygid, posix_ctermid, posix_getcwd, posix_getegid, posix_geteuid, posix_getgid, posix_getpgrp, posix_getuid, posix_getpgrp, posix_getppid, posix_getsid, posix_isatty, posix_mknod, posix_mkfifo, posix_setegid, posix_seteuid, posix_setgid, posix_setpgid, posix_setsid, posix_setuid, posix_times, posix_gethostname, posix_uname, posix_uname, proc_get_status
suhosin.mail.protect = 2
suhosin.memory.limit = 64M
suhosin.get.max_name_length = 128
suhosin.get.max_totalname_length = 300
suhosin.get.max_value_length = 1526
suhosin.get.max_vars = 150
suhosin.post.max_array_depth = 256
suhosin.post.max_array_index_length = 2048
suhosin.post.max_name_length = 512
suhosin.post.max_totalname_length = 2048
suhosin.post.max_value_length = 256000
suhosin.post.max_vars = 1526
suhosin.post.disallow_nul = on
suhosin.request.max_totalname_length = 300
suhosin.request.max_vars = 250
suhosin.request.max_varname_length = 128

code/suhosin.ini

Source: Author, only adjusted configuration options shown.
<VirtualHost ::80 ff06::c3:80>
  ServerName domain.cz
  ServerAlias * domain.cz
  ServerAdmin abuse@domain.cz
  DocumentRoot /home/www/domain.cz/web/

  # PHP settings
  php_admin_value upload_tmp_dir /home/www/domain.cz/php_upload
  php_admin_value session.save_path /home/www/domain.cz/tmp

  # Overrides
  AccessFileName .htaccess
  <Directory />
    Options FollowSymLinks Indexes
    AllowOverride AuthConfig FileInfo Indexes Limit
  </Directory>
  DirectoryIndex index.html index.htm index.php index.php5 index.php4
  index.php3 index.shtml index.cgi index.pl index.jsp Default.htm Default.h tm

  # Subdomains
  RewriteEngine on
  RewriteCond %{HTTP_HOST} !^www\.. domain\..cz
  RewriteCond %{HTTP_HOST} ^(\[\|\])\.. domain\..cz
  RewriteRule ^(.*) /%1/$1 [L]
  RewriteCond %{HTTP_HOST} ^www\.. domain\..cz [OR]
  RewriteCond %{HTTP_HOST} ^domain\.. cz [OR]
  RewriteRule ^(.*) /home/www/domain.cz/web/www/$1 [L]

  # Logs
  ErrorLog /home/www/domain.cz/logs/error_log
  LogLevel warn
</VirtualHost>

code/vhost

Source: Author.
Appendix E

Zen Cart – session tokens

Figure E.1: Zen Cart – session token regeneration (step 1: redirect from unencrypted to encrypted page, a new token sent as hidden form value).

http://domain.cz/zencart/admin/
GET /zencart/admin/ HTTP/1.1
Host: domain.cz

HTTP/1.1 302 Found
Set-Cookie: zenAdminID=559e2a8613580a996969c12aaed0d2bd; path=/zencart/; domain=.domain.cz
Location: https://domain.cz/zencart/admin/login.php?zenAdminID=559e2a8613580a996969c12aaed0d2bd

https://domain.cz/zencart/admin/login.php?zenAdminID=559e2a8613580a996969c12aaed0d2bd
GET /zencart/admin/login.php?zenAdminID=559e2a8613580a996969c12aaed0d2bd HTTP/1.1
Host: domain.cz
Cookie: zenAdminID=559e2a8613580a996969c12aaed0d2bd

HTTP/1.1 200 OK
Content-Type: text/html; charset=iso-8859-1

code/zen_session_tokens1.txt

Source: Author.
Figure E.2: Zen Cart – session token regeneration (step 2: login form submitted, new cookie is assigned – completely over a secure channel).

https://domain.cz/zencart/admin/login.php
POST /zencart/admin/login.php HTTP/1.1
Cookie: zenAdminID=559e2a8613580a996969c12aad0d2bd
Content-Type: application/x-www-form-urlencoded
Content-Length: 91
admin_name=bp&admin_pass=654789&securityToken=eed9f703cf4f2d2ba91e966d8362c9b79ksubmit=Login

HTTP/1.1 302 Found
Set-Cookie: zenAdminID=eec6197e6028b8381524c4850f8ae9; path=/zencart/
admin; domain=.domain.cz
Location: https://domain.cz/zencart/admin/index.php

GET /zencart/admin/index.php HTTP/1.1
Host: domain.cz
Referer: https://domain.cz/zencart/admin/login.php
Cookie: zenAdminID=eec6197e6028b8381524c4850f8ae9

HTTP/1.1 200 OK
Content-Type: text/html; charset=iso-8859-1

code/zen_session_tokens2.txt

Source: Author.
## Appendix F

### Payment methods

Table F.1: Payment methods used when shopping on line, Germany

<table>
<thead>
<tr>
<th>Payment method</th>
<th>% of respondents (multiple answers possible)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mobile phone</td>
<td>3.3</td>
</tr>
<tr>
<td>Prepaid systems</td>
<td>22.6</td>
</tr>
<tr>
<td>Billing systems</td>
<td>54.9</td>
</tr>
<tr>
<td>Credit card</td>
<td>62.4</td>
</tr>
<tr>
<td>E-mail</td>
<td>14.4</td>
</tr>
<tr>
<td>Online-debit</td>
<td>70.3</td>
</tr>
<tr>
<td>Online-(giro)-transfer</td>
<td>79</td>
</tr>
<tr>
<td>Paper-based debt</td>
<td>40.6</td>
</tr>
<tr>
<td>Paper-based (giro)-transfer</td>
<td>53.3</td>
</tr>
<tr>
<td>Cash on delivery</td>
<td>63.9</td>
</tr>
</tbody>
</table>

Source: [26, pg. 14].
Appendix G

Survey screenshots

Figure G.1: Facebook – non-SSL login page

Source: Author.
Figure G.2: Facebook – SSL login page

Source: Author.

Figure G.3: Firefox 3 – self-signed certificate warning

Source: Author.
Figure G.4: Altered web page with malicious code possibly inserted (beginning of *Troj/JSRedir-R (Gumblar))*

Source: Author.
Appendix H

Content of Enclosed DVD

There is a DVD enclosed to this thesis which contains source codes, screenshots, server configuration and server logs.

- Folder `source_code`: source codes of evaluated software packages.
- Folder `survey`: screenshots of the survey and answers of the participants.
- Folder `logs`: server logs (initial honey-pot study).
- Folder `config`: server configuration.
- Folder `stats`: visitor statistics used for browser share estimates and others.