The major reason why most people are still sceptical about e-commerce is the perceived security and privacy risks associated with e-transactions, e.g., data, smart cards, credit cards and exchange of business information by means of online transactions. Today, vendors of e-commerce systems have relied solely on secure transaction protocols such as SSL, while ignoring the security of server and client software. This article, Secure Business Application Logic for e-commerce Systems, discusses a key weak link in e-commerce systems: the business application logic. Although the security issues of the front-end and back-end software systems in e-commerce application warrant equal attention, but this research focuses on the Security of Middle Tier of e-commerce server that implements the business application logic. Although the security issues of the front-end and back-end software systems in e-commerce application warrant equal attention, but this research focuses on the Security of Middle Tier of e-commerce server that implements the business application logic and traditionally, e-commerce sites implemented the middle tier of software on the web server using CGI. We also present strategies for secure business application logic: good design and engineering, secure configuration, defensive programming and secure wrappers for server-side software.

© 2004 Elsevier Ltd. All rights reserved.

Introduction

A risk constitutes something negative or bad. Business people, however, need to think of risks as mere opportunities, the reason being that, in most business environment, the number or size of the risks taken usually is equal to the number or size of the advantages to be gained. The reverse is also true. Namely, the number or size of the risks taken usually also equals the losses potentially to be sustained. It is, therefore, important always to take calculated risks and to be aware of all possible consequences (Labuschagne and Eloff, 2000).

The advent of the electronic commerce ushered in a new period pervaded by a sense of boundless excitement and opportunities. Although it took some time, but now most organisations have already involved conducting electronic commerce as that world wide e-Commerce Market has
“Electronic Commerce”, also referred to as “e-commerce”, has revolutionised the modern-day business world. Thanks to its concomitant technologies, new business opportunities have been created. This results in the survival or downfall of many organisations on the global economic playing field, depending on whether they choose to seize or fail to avail themselves of these opportunities. The new opportunities, however, come with their own set of problems. The major concern cited by most decision-makers when it comes to e-commerce, is “Security, Privacy and Client Trust”. For this reason, subscribers still feel uncomfortable about the idea of trading on the Internet (Foresight, 1998).

To them, the possible risks to be incurred justify the potential rewards. Unfortunately, their fears are not unfounded because of trust, privacy and security threats to all e-commerce transactions. Basically, e-commerce is concerned with doing business using electronic technologies. It can involve the transaction of data, transaction of payments or marketing information, and value addition to existing products or databases. E-commerce can involve the exchange of credit card numbers that represents purchases by a consumer from a retailer (Lawrence and Corbett, 2000).

Credit card is one of the primary means of electronic payment on the Internet. Although a large number of users reported that they had their credit card stolen, there is still a lot of consumer confidence in the use of credit card. Again, this trust should not be betrayed and arrangements should be made to assure that consumer confidence remains strong in credit card specially for those who are reluctant (Kalakota and Whinston, 1996).

According to the survey report by Bank of England on 15-10-2002, this year the credit card theft worth was four hundred and eleven million pounds (fraud of credit cards), which was a 50% increase compared to the previous year in the UK (Survey Report by Bank of England, 2002).

Another case reported in 2003 where the number of user of Barclay Bank suffered while conducting online credit card transaction this time was the case of during the transaction client to server side, actually it was the case of "Web Copy Cat Trick". This technique is the cause of mis-configuration server side components that provided path way to intruder to play a technique which is known as "Web Copy Cat Trick" and organisation (Barclay Bank) suffered heavy losses due to that, they had to advise their customers not to use online services until problem is resolved.

Another cyber crime case reported by MSNBC news report: the issue of international credit card thievery and fraud burst into the public consciousness in January 2004 with news of a heist of thousands of credit cards from the CD Universe Web site, allegedly by a teenage hacker from Russia. It continued to make headlines with news that Amazon.com had uncovered a Russia-based plot to defraud it and other e-merchants out of more than $70,000 worth of merchandise using stolen credit-card information.

Stephen Orfei, vice president for electronic commerce and emerging technology at MasterCard, said that the Internet accounts for between 2 and 2.5 percent of total credit card transactions. Using the lower figure and applying it to the bank association’s total fraud losses of $526 million in 1998 yields a loss to online fraud of slightly more than $10.5 million.

Visa USA reported $487 million in fraudulent charges last year, which assuming a 2 percent rate would be approximately $9.75 million.

American Express and Discover Financial Services do not publicly report fraud rates and declined to comment on the subject of Internet credit card fraud except to say that they aggressively attempt to comb. Since e-commerce fraud rates are one-third higher than overall fraud rates, according to Visa’s figures for 1999 (9 cents per $100 on the Net vs. 6 cents per $100 overall), that $20 million figure for Visa and MasterCard alone is most likely on the low side. The federal government’s main mechanism for tracking economic crimes is the Treasury Department’s Financial Crimes Enforcement Network, known as FinCen, which collects reports on 17 types of “suspicious activities” from U.S. banks and other institutions, the agency received about 120,000 reports, of which 4900 — or about 4 percent — reflected possible credit card fraud. Increasing ratio of e-commerce fraud rates is real concern.

**Concerns of e-commerce**

Any business that is still in business, conducts “commerce”. Commerce is the exchange of money for goods or services between companies (B2B) or end consumers (B2C). “E-commerce” is doing commerce using electronic technology such as intranets, extranets and the Internet, which
provides with a new means of obtaining useful information and purchasing products and services between companies and end consumers. Although this form of e-commerce has undergone rapid growth, particularly through the use of the Internet, business and consumer fears and concerns about the risks, both have inhibited its growth real and perceived, of doing business electronically (Kamthan, 1999).

From the time a business installs a web server or hires space on a commercial web server from an ISP, there is a potential for business systems in the organisation to be exposed to breaches of security and confidentiality across the entire Internet (Lawrence and Corbett, 2000).

Any link to the Internet exposes the business to tampering, or Internet graffiti, where data can be exposed with meaningless scribble, pictures or electronic junk in the same way that graffiti artist scrawls on walls. Link to the Internet also exposes the business to the theft of data. Database can be very easily captured wholly and transferred for other uses such as industrial espionage (Sielgel, 1996).

The TCP/IP protocol developed to run the Internet was not designed with security in mind. This protocol, the basic system running Internet communications, is vulnerable to interceptions (Hunt, 1998).

Any movement of data from a browser to a server or back is vulnerable to eavesdropping (Chaffy, 2000). Website security is about keeping strangers out, but at the same time allowing controlled access to the network (Lawrence and Corbett, 2000).

Therefore, due to all these reasons, newer schemes for security, such as Secure Socket Layer (SSL), have been introduced. This is a low-level encryption scheme used to encrypt e-commerce communications and for consumers to use their credit card numbers in transactions on a secure WWW form and transmit the form over the Internet without the risk of a cracker obtaining the credit-card information in higher-level protocols such as SHTTP, NNTP, and FTP.

The SSL protocol can authenticate server (i.e., verify server’s identity) encrypting data in transit between the server and client accessing the server (Ahuja, 1997).

Basically, cryptography is based on encryption, which use the authentication and digital signature techniques based on Cryptographic Methods. Encryption is the transformation of data into a form unreadable by anyone without a secret key. Its purpose is to ensure privacy by keeping the information hidden from anyone for whom it is not intended, even those who can see the encryption data. There are two types of cryptographic systems: (1) symmetric system, where a single key serves both as the encryption and decryption; (2) asymmetric system, where each person gets a pair of keys called the public and private key. Each person’s public key is published, while the private key is kept secret. The need for the sender and receiver to share secret information is eliminated. All communications involve only public keys, and no private key is ever transmitted or shared. Furthermore, public and private key cryptography can be used for authentication using Cryptographic Envelop Technology (which is developed in Java Archive) technique (Lawrence and Corbett, 2000). This session key is used to encrypt the HTTP transaction (Furche and Wrightson, 1996).

Nevertheless, the advancement of the security field has proved that vendors of e-commerce systems cannot solely rely on secure transaction protocols, such as SSL – an encryption protocol promoted as proof of 100% security by e-commerce vendors. Lost in the hype are the real security risks of e-commerce and these attributes make up only a small part of security, privacy and client trust of e-commerce risks.

In what follows, we will discuss further about the advancement on security, privacy and client trust of e-commerce risks, making secure business application logic for e-commerce security, privacy and client trust in which we will describe about e-commerce security more than cryptography and the idea of secure business application logic for e-commerce.

**Further advancements in the field**

**E-commerce security; more than cryptography**

As the market for e-commerce continues to heat up significantly, most stakeholders in the market are quick to declare that the system is secure. Encryption protocols such as SSL are promoted as proof of security by e-commerce vendors. Lost in the hype are the real security risks of e-commerce. Although encryption of data during transactions provides crucial confidentiality, integrity, and authentication, these attributes make up only a small portion of security that must be ensured for e-commerce security. Businesses engaging in e-commerce incur higher risk of attacks, higher risk of severe losses, and a higher loss-to-incident ratio than simply put up web pages. In fact, due to
the complex nature of business, e-businesses are much more vulnerable than simple websites due to the greater complexity of the software necessary to support e-commerce transactions (Garfinkel and Stafford, 1997; Gosh, 1998).

The software that executes on either end of the transaction — sever-side or client-side software — poses real threats to the security, privacy and client trust of all e-commerce transactions. Two familiar adages play an important role in understanding how to secure e-commerce systems: (1) a chain is only as strong as its weakest link; and (2) in the presence of obstacles, the path of least resistance is always the path of choice (Ghosh, 2000). Encryption protocols generally provide the strongest perceived security in all the components involved in e-commerce transaction (Garfinkel and Stafford, 1997).

However, in e-commerce as in other real-world systems, the security of the system is only as strong as its weakest component. On the whole, the security of server-side systems is much weaker than the security provided by secure data transaction protocols such as SSL, example of this claim refers to the above mentioned cases of security breach and the case study of Barclay Bank Web site targeted that is the case of Copy Cat Trick which is conducted through the server-side misconfiguration of softwares, cause of vulnerability/compromising server-side security is clearly proof of this claim. Considering these two adages together in the context of e-commerce security, privacy and client trust, it becomes clear that malicious perpetrators will rarely attempt to break encryption codes when they can much more easily break into a system, and enjoy a much higher return via this simpler approach (Ghosh, 2000).

This article addresses the topic of securing server-side software used in e-commerce applications. Today, most e-commerce server applications are implemented in n-tier architecture as shown in Fig. 1. This architecture usually consists of a front-end web server, a business application layer, a back-end database, ERP systems, and legacy software. In practice, many of the security issues in e-commerce sites are specific to the way in which these components are configured

### Expecting the unexpected

While developers can write CGI scripts in any general-purpose programming language, they most often write such scripts in Perl, C, Tcl, or Python. The basic rule of thumb when designing CGI scripts is to expect the unexpected. Or more appropriately, expect the malicious. While developers have control over the content of CGI scripts, they have no control over what users will send to the CGI scripts. Also, do not overlook attacks against CGI scripts that exist on the server as part of the distribution, but are not even used as part of the e-commerce application. Some CGI scripts included as part of the web server distribution have well-known flaws that can be exploited to obtain unauthorised access to the server. Even if developers do not use the default CGI scripts as part of the web server pages, anyone else can use them by simply knowing the script names (Stein, 1998).

More recently, component-based software (CBS) has made inroads in e-commerce applications. The idea of using CBS is to develop, purchase, and reuse industrial-strength software in order to rapidly prototype business application logic. One of the more popular component frameworks for e-commerce applications is Enterprise Java Beans.
(EJB), which supports component-based Java. Other component models include the Common Object Request Broker Architecture (CORBA), an open standard developed by the Object Management Group (OMG), and Microsoft’s Common Object Model (COM) and Distributed COM (DCOM). The component frameworks are the “glue” that enables software components to provide services such as business application logic and use standard infrastructure services such as naming, persistence, introspection, and event-handling, while hiding the details of the implementation by using well-defined interfaces (Gosh, 1998).

Fig. 1 shows typical n-tier architecture for a component-based e-commerce application. The web browser client can display web pages in HTML, execute mobile code such as Java applets or web scripts, and capture business-specific semantics using XML. The web server provides web services in addition to other Internet services such as e-mail and File Transfer Protocol (FTP). The business application logic is coded in software components that can be custom-developed or purchased off-the-shelf. The application servers provide the infrastructural services for particular component models such as EJB, CORBA, COM, and DCOM. They also provide an interface for the business application logic to back-end services such as database management, enterprise resource planning (ERP), and legacy software system services.

Component-based software for business-to-business applications

In addition to supporting traditional CGI functions, component-based software is expected to enable distributed business-to-business applications over the Internet. The component-based software paradigm also supports good software engineering (see “Designing for Security” section). The Unified Modelling Language (UML) supports object-oriented analysis and design for component-based frameworks. In addition, as the market for component-based software heats up, many standard business application logic components will be available for purchase off the shelf.

The security risks of component-based software

Although component-based software provides numerous benefits, it poses security hazards similar to CGI scripts. Component-based software enables software development in general-purpose
programming languages such as Java, C, and C++. As such, these components execute with all the rights and privileges of server processes. Like CGI, they process untrusted user input. However, because component-based software can be used to build sophisticated large-scale applications, errors are unarguably more likely with component-based software than with simple CGI scripts. Regardless of the implementation — CGI or application servers — the security risks of server-side software are high, and therefore server-side software must be carefully designed and implemented using the techniques covered in the following sections (Stein, 1998).

Designing for security

As with all software development, good design and engineering practices are important for software quality. This point is particularly important for development of security-critical software such as e-commerce applications. Rather than thinking of security as an add-on feature to software systems, security should be designed into the system from the earliest stages of requirements gathering through development, testing, integration, and deployment. The goal of design for security is to break the penetrate-and-patch mindset that pervades commercial software security today, and replace it with a process for finding and removing security-related bugs prior to software release. Finding and fixing bugs in software after release costs orders of magnitude more than correcting them early during the software development lifecycle (Gosh, 1998).

One unfortunate consequence of the huge pressure to reduce time to market for e-commerce services is that good software engineering practices are dropped (or more often never started). As described in section "A key weak link: business application logic", the demand for e-commerce applications is driving the complexity of business application logic. Developing software such as a typical desktop application for one user at a time differs significantly from developing an e-commerce application that needs to handle tens of thousands of concurrent requests. Today's e-commerce applications need to not only handle many simultaneous users, but also deal with malicious threats. As a result, developers must employ good software engineering practices to develop robust and secure business application logic. Several key activities mark good software engineering practices:

- Gathering and formally specifying requirements
- Developing normal and pathological usage scenarios
- Object-oriented analysis and design
- Adopting good coding practices
- Unit testing and integration
- Release engineering
- Third-party validation and verification

Requirements' gathering has traditionally focused on users' needs for the application under development. In gathering requirements for security-critical applications, developers must equally focus on what the user should be allowed to do and should not be allowed to do. A plan for meeting functional requirements is typically captured in specifications for the application. However, specifications for e-commerce applications need to specify not only which functional behaviour is expected from the application, but also which behaviour is not desired.

Therefore, to develop a specification of secure behaviour, the project team must develop a security policy for the e-commerce application (and its users). Developing specifications is often fraught with anxiety over formality. While formal notations can be useful for reducing ambiguity, if they are so daunting that they discourage the development of specifications, then they can have the opposite effect on software quality. Remember that the ultimate goal of software specification is to promote understanding of intended system behaviour, so any specification is better than none. A specification of undesired behaviour is not only useful for preventing security design flaws, but can also be used for security-oriented testing (Gosh, 1998).

Configuring the CGI

One of the most common — yet easily preventable — security hazards is misconfiguration of software. CGI scripts, too, must be correctly configured for security. One feature supported by many web servers is the ability for individuals throughout an organisation to write CGI scripts and have them executed from their own directories. Although useful for "sniffing" personal web pages, such scripts can introduce system security hazards. In e-commerce applications, the web server should be configured to prevent CGI scripts from executing anywhere but a single CGI directory controlled by the system administrator. The script-aliased CGI mode for web servers ensures that CGI scripts will
execute only from an explicitly named directory in the server configuration file. In addition, the CGI script path is not named in the URL to the CGI. Rather, the server "aliases" the explicit path to the CGI scripts to a name of choice such as "cgi-bin". Thus, running the server in script-aliased CGI mode prevents rogue CGI scripts from executing, and also hides the explicit path to the CGI scripts (Pre-Built CGI Scripts).

Configure the CGI script directories correctly using operating system access controls. For instance, if CGI scripts are written in a compiled language (such as C), the script sources should be excluded from the document root of the web server, so that they cannot be accessed via the web. They should be accessible to the system administrator or web content development group only, and inaccessible to everyone else in the organisation. If the script sources fall into the hands of malicious perpetrators, the source code can be inspected for flaws, making the perpetrators' job even easier. Access to the CGI executables directory, frequently called the cgi-bin, should also be properly controlled. Only the web server and administrator need access to this directory. Liberal access permissions to the CGI executables directory let malicious insiders place their own scripts on the e-business site. Also, be sure to prohibit access to interpreters from the web server. For example, system administrators may be tempted to include the Perl interpreter in CGI script directories. However, doing so provides direct web access to interactively execute Perl commands — an extremely dangerous configuration.

Finally, account for every CGI program on the server in terms of its purpose, origin, and modifications. Remove CGI scripts that do not serve a business function. View with suspicion and carefully screen CGI scripts that are distributed with web servers, downloaded from the Internet, or purchased from third parties. These steps will eliminate most of the potentially dangerous CGI scripts. Once a stable set of CGI programs has been established, make a digital hash of the executable programs (using MD5) to enable future integrity checks (Christiansen and Torkington, 1998).

**Defensive programming**

Simply configuring the system securely is not enough to ensure security. Even after the CGI scripts are properly locked down, flaws in the design and implementation of the CGI scripts can be exploited to leverage full system compromise. A good starting place for writing secure CGI scripts is to use safe language features. For instance, although interpreted languages such as Perl are great for rapidly prototyping software, they also provide powerful constructs for executing system commands that perpetrators can easily exploit. Perl commands such as `eval()`, `system()`, backquotes ('`'), pipes, and `exec()` can potentially result in system commands being executed on the web server at the discretion of an unknown and untrusted remote user. Use these commands with extreme caution when processing user input.

Although the same types of system commands can be used in compiled languages such as C, they are not as easily constructed or misused as system commands in interpreted languages such as Perl, Python, or Tcl. However, C provides its own hazards due to its lack of advanced memory management. For instance, the most common and notorious security-related bug for C programs is the buffer overflow. This condition occurs when the memory allocated to a variable is less than the amount being written into it. Regardless of the language used for implementing the business application logic, the Cardinal's rule in development is never to trust the user input.

**Sanitising user input**

Fortunately, defensive programming can add security where powerful language constructs might otherwise fall short. A key defensive strategy is to sanity-check (and sanitise) all program input. For instance, if a social security number is requested, be sure that the input is numeric and that no more than nine characters are read. Also be sure to sanitise input of special Meta escape characters. Characters such as the backtick (``) allow input to be executed as commands to the interpreter. Unescape special characters, or if possible, accept only legitimate characters. Do not assume that the length of the input is fixed. Even if the length of the input field is limited in a form, the user is not constrained to using the form to send in user input. When parsing user input, read only the length of data necessary for the input and ensure that adequate memory is allocated for the amount of input to be read (Ghosh, 2000).

**Using environment variables**

A common oversight in CGI script development is the use of environment variables. The user can set certain environment variables. Sanitise or sanity-check environment variables from HTTP clients before using them. Use other system environment
variables with caution. Specify explicit paths rather than rely on system environment variables for accessing programs and files. An insider can fool a CGI script that depends on an environment variable to execute a rogue program instead of the intended program. Also, do not use hidden fields in forms in order to hide information. Though hidden fields may not display in the browser, they are easily discerned by inspecting the HTML source.

Leveraging existing safety features

In addition to these defensive programming techniques, some languages provide safety features that should be leveraged when possible. For instance, Perl 5 contains a "taint" module for preventing user-supplied input from being used in system commands. When Perl tainting is enabled, any variable derived from user input (such as command-line arguments and environment variables) is considered tainted (or untrusted). Thus, when a tainted variable is used in a system command, such as modifying the file system or spawning a shell, the Perl runtime interpreter prevents the command from executing.

Similarly, Tcl has a safe language interpreter that can prevent untrusted input from compromising system security and integrity. When using application servers for implementing business application logic, prefer using strongly typed languages such as Java rather than C or C++. Java's type safety and advanced memory management can eliminate typical programmer errors in walking through memory that often lead to security breakdowns. Furthermore, Java 2 provides fine-grained access control for Java programs to prevent unauthorised access to system resources.

In summary, always expect the unexpected from user input. Employ defensive programming techniques such as sanitising input regardless of the language used. If you develop in an interpreted language, then use the safe language-interpreter features such as Perl tainting and Safe-Tcl. When developing business application logic from components, use a type-safe language such as Java rather than languages such as C and C++, which give free reign to the programmer to use memory objects in any context.

Wrapping the business application logic

The defensive programming techniques described in the previous section can significantly help to write secure server-side software. Even so, perfection is largely unattainable in software development. To improve the security of the business application logic, consider using one of several existing technologies that can "wrap" server-side software in order to limit potential damage from flawed software exploited for malicious gain.

One of the most commonly employed approaches on UNIX-based systems is to use the chroot () command to effectively cordon off the file system visible to a program. The chroot () function changes the effective root of the file system for a given process to a designated new root. For server-side software, define the new root as the smallest partition of the file system necessary for the programs to access. When a process is run in a chroot () environment, it is effectively running in a jail cell to prevent it from causing damage to other portions of the system. While the chroot () environment is useful for preventing auxiliary damage to the rest of the system, it is important to remember that the chroot () environment must contain the portion of the file system necessary for the program to perform its function.

If the business application logic needs to communicate with critical system resources such as back-end databases, then these resources need to be included within the chroot () environment. In this case, the chroot () environment will not protect databases or other included resources from misbehaving software, but will protect the integrity of other system resources outside the chroot () environment. Alternative commercial technologies for "sandboxing" executing processes are available on the Windows platform.

Another technique for UNIX-based systems provides some protection by running CGI scripts with user permissions. The program CGIWrap is designed for systems that allow internal users to write and post their own CGI scripts. Although allowing users to post their own CGI scripts is not advisable for e-commerce applications, if it is a requirement, then CGIWrap can be useful for limiting the privilege of the executing CGI process to that of the user. The technique assumes that permissions of users are properly configured on the system such that no user is allowed access to other users' files or to critical system files and programs.

In many systems, the web server is configured to run all CGI scripts under a single account's privileges. Usually an account is created for this purpose, such as www or nobody. However, in systems where internal users are allowed to post their own CGI scripts, it is not advisable to allow a user's script to access files owned by other users' or other web server files. CGIWrap enforces the policy that users' CGI scripts can only access files
owned by the script’s owner. Again, before leaving the discussion of CGIWrap, it is important to reiterate that it is generally not advisable to allow users to post their own CGI scripts to an e-commerce site.

A more attractive alternative to CGIWrap is the SBOX tool written by Stein (1998). Like CGIWrap, SBOX is designed for multi-user websites where users are allowed to post their own CGI scripts. However, in addition to limiting the privilege of CGI scripts to their owner’s files, SBOX can also place limits on executing CGI processes on the use of system resources such as CPU cycles, memory, and disk. This feature provides an effective way to thwart denial-of-service attacks from malicious CGI scripts, or poorly written CGI scripts that end up consuming system resources to the detriment of the other e-commerce services. SBOX also allows CGI scripts to be run in a chroot () directory. Thus, SBOX provides wrapping capability of programs that not only limits the file system using chroot (), but also ensures that the CGI scripts cannot infringe on other users’ programs and will not deny service to other executing processes.

It is important to stress again; however, that e-commerce applications should not be deployed in a general-purpose environment that allows multiple users to upload their own CGI scripts. But if CGI uploading does need to be supported, tools such as SBOX and CGIWrap can reduce the likelihood of catastrophic security breaches.

Conclusion

E-commerce security, privacy and client trust issues should always be dealt with when surfing on the web or using the Internet for business purposes.

E-commerce encompasses all aspects of using the Internet for business or personal use. Now, more than ever, a great deal of business is performed in one-way or another over the Internet. For some, it is simply ease of communication; for others, having the ability to research topics, products, or even people make the Internet an absolute necessity for business.

Businesses have begun exploiting the Internet for commercial transactions. Recognizing the dangers in sending confidential information over an inherently insecure media, a number of secure data transport protocols have emerged. At minimum, these protocols encrypt sensitive information such as credit card numbers to prevent unauthorised people from capturing the data. Some protocols even facilitate payment for merchants through banking institutions.

Even with the strong security provided in the transport of data, e-commerce security still remains elusive. In practice, most security violations occur through other avenues than breaking cipher text. Using encryption on the Internet is the equivalent of arranging an armoured car to deliver credit-card information from someone living in a cardboard box to someone living on a park bench. The point is that usually we infer security from encryption when we are so vulnerable otherwise.

Therefore, keeping in view of strategy of developing secure business application logic for e-commerce sites (server-side security and privacy) can secure from breaching by securing server-side software, because security is as strong as its weakest link. This is why common sense is the best tool to make your side secure from hacking/breaching system.

Common sense is the appropriate tool that can be used to establish your security policy. Elaborate security schemes and mechanisms are impressive and they do have their own reasons, but yet there is little point in investing money and time on an elaborate implementation scheme if the simple controls are forgotten.

References

Chaffy D. Business information systems; technology, development and management in the e-business; 2000. ISBN: 027365540X.
Ernst, Young. Internet shopping – an Ernst & Young special report; 1998 [Section 2].
Labuschagne L, Eloff JHP. E-commerce: the information-security challenges journal. Information Management &
Faisal Nabi, Pakistani Scientist in the field of Computer Science Security, was born in 1978 in Karachi. He had his initial education from Aisha Bawany school and college education from Govt Allama Iqbal. He then joined Karachi University for graduation, post-grad level. He then went to UK for higher education, joined MSc e-commerce/e-business at University of Luton. He specialized in information/e-commerce security. Specialities: cryptography, stegnography, virtual disk design with blue fish encryption technique, designing invincible security architecture. Started research as R&D researcher at the University of Luton without any support by the University and other assistance. Research conducted on self-finance suffered all research expenditures. In April 2004, he was appointed as Researcher at Deveraux & Deloitte Research Centre, UK. I want to say special thanks to Dr. Carsten Maple Department of Computing and Information Systems, University of Luton who always encouraged me time to time, he is a top class Researcher.