

ABSTRACT

This paper stresses the need for effective security testing by filling up gaps caused by insecure communication protocols. Application security poses one of the greatest challenges to any development team.

The paper talks about how web application frameworks are vulnerable to insecure communications. No matter how well an application has been designed and developed, due to its complex nature it will contain some flaws that will surface after a certain amount of time. These errors can be caused due to large volumes of code, complex internal interactions, interoperability with uncertain external components and unknown interdependencies coupled with vendor cost and schedule pressures.

INTRODUCTION

Most people today do not understand what security is or is not about as evidenced by so many works of modern fiction centering around a plot where the terrorists/foreign government/aliens "bug" a server, a cable, or a satellite. Today's technology is supposed to prevent attacks involving any layer in the middle being bugged. Besides not understanding what modern security is capable of, many who are working with it do not understand what it is not capable of.

"Security as it is today is an illusion".

It is shocking the sheer number of online services or applications one can install (forums, webmail, blog, wiki, etc...) that have insecure logins. Nearly all of them take user login credentials in plain text, allowing anyone between the user's computer and the website's application to steal the passwords.

Insecure communication results when a client and server communicate over a non-secure (non-encrypted) channel. Failing to securely communicate server-to-server and server-to-client helps attackers to intercept sensitive transactions. Attackers do this by using man-in-the-middle attacks, a post for another time. Not communicating securely breaks down confidentiality and integrity.

Anyone can fall prey to insecure communication while using the following:

- Insecure client-to-server connections.
- Insecure their server-to-database connections.
- Insecure other back end connections that pass sensitive data.

Security testing has recently moved beyond the realm of network port scanning to include probing software behavior as a critical aspect of system behavior. Unfortunately, testing software security is a widely misunderstood task. Security testing done properly goes deeper than simple black-box probing on the presentation layer (the sort performed by so-called application security tools) and even beyond the functional testing of security apparatus.

Testers must use a risk-based approach, grounded in both the system's architectural reality and the attacker's mindset, to gauge software security adequately. By identifying risks in the system and creating tests driven by those risks, a software security tester can properly focus on areas of code in which an attack is likely to succeed. This approach provides a higher level of software security assurance than possible with classical black-box testing.

Accordingly, security testing must fill the gap in system development and actual operation of these systems. Organizations that have an organized, systematic, comprehensive, on-going, and priority driven security testing regimen are in a much better position to make prudent investments to enhance the security posture of their systems.

Web applications galore and are difficult to run effectively without the added efficiency and communications brought about by the Internet. At the same time, the Internet has brought about problems as the result of intruder attacks, both manual and automated, which can cost many organizations excessive amounts of money in damages and lost efficiency.

Thus, organizations need to find methods for achieving their mission goals in using the Internet and at the same time keeping their Internet sites secure from attack. Computer systems today are more powerful and more reliable than in the past; however they are also more difficult to manage.

Understanding Insecure Communication and Security Testing

Secure Communication is about making communication behave in the presence of a malicious attack, even though in the real world, communication failures usually happen spontaneously that is, without intentional mischief. Not surprisingly, standard software testing literature is only concerned with what happens when software fails, regardless of intent. The difference between software safety and software security is therefore the presence of an intelligent adversary bent on breaking the system.

Security is always relative to the information and services being protected, the skills and resources of adversaries, and the costs of potential assurance remedies; security is an exercise in risk management.

Applications frequently fail to encrypt network traffic when it is necessary to protect sensitive communications. Encryption (usually SSL) must be used for all authenticated connections, especially Internet-accessible web pages, but backend connections as well. Otherwise, the application will expose an authentication or session token. In addition, encryption should be used whenever sensitive data, such as credit card or health information is transmitted. Applications that fall back or can be forced out of an encryption mode can be abused by attackers.

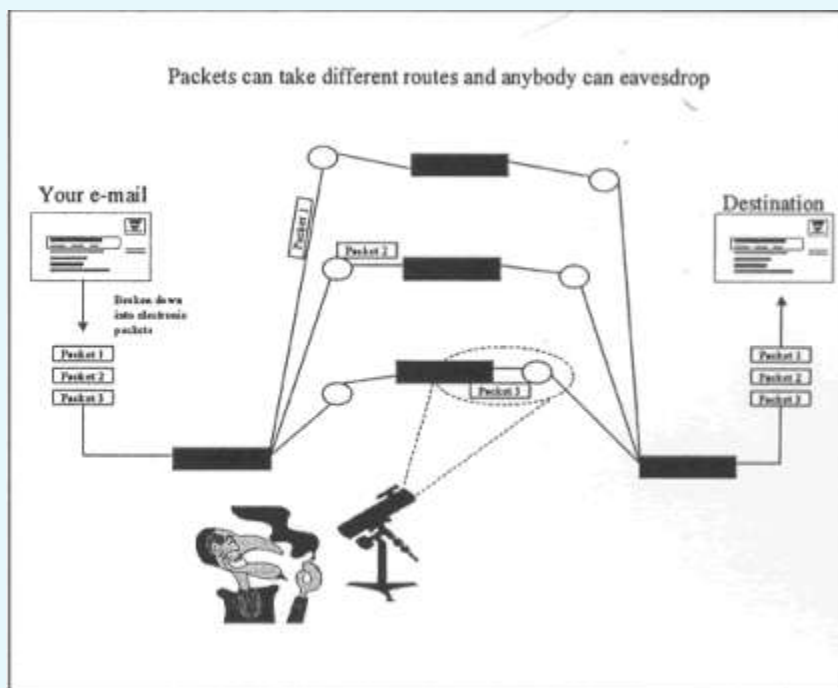


Figure 1

Failure to encrypt sensitive communications means that an attacker who can sniff traffic from the network will be able to access the conversation, including any credentials or sensitive information transmitted. Consider that different networks will be more or less susceptible to sniffing. However, it is important to realize that eventually a host will be compromised on almost every network, and attackers will quickly install a sniffer to capture the credentials of other systems.

Encrypting sensitive data, such as credit cards and social security numbers, has become a privacy and financial regulation for many organizations. Neglecting to use SSL for connections handling such data creates a compliance risk.

Protection against Vulnerability

Organizations need to find methods for achieving their mission goals in using the Internet and at the same time keeping their Internet sites secure from attack. Sending information over the internet, whilst preserving its integrity is becoming a must. However information sent over insecure channels makes it vulnerable to sniffing, or manipulation whilst in transit. This leads to majority of web application frameworks vulnerable to insecure communication.

As internet communication is primarily based on TCP/IP protocols (operating at the transport layer of the OSI model), and takes place in plain text, thus making it easily interceptable. To make matters worse, traffic sniffing tools are widely available which facilitate packet sniffing in transit.

The different technologies available to ensure secure communication are:

- Secure Sockets Layer (SSL)
- Secure Electronic Transaction (SET)
- Secure HTTP (S-HTTP)
- Secure Shell (SSH)
- IPSec

Payment Protocols (SET, CyberCash, First Virtual)			
S-HTTP	HTTP	S/MIME	Telnet, mail, news, ftp, dns and others
Secure Socket Layer (SSL) (provides secure connection)			
Transport Control Protocol (provides reliable delivery of packets)			
Internet Protocol (routing of packets)			
Data Link Layer			

Secure Sockets Layer (SSL)

Using SSL for communications with end users is critical, as they are very likely to be using insecure networks to access applications. Because HTTP includes authentication credentials or a session token with every single request, all authenticated traffic needs to go over SSL, not just the actual login request.

Encrypting communications with backend servers is also important. Although these networks are likely to be more secure, the information and credentials they carry is more sensitive and more extensive. Therefore using SSL on the backend is quite important.

SSL is the most widely deployed protocol for securing HTTP communication, and works above the 4th layer of the OSI model. HTTP when deployed over SSL or Transport Layer Security (TLS) is popularly known as HTTPS. Testing for security on most popularly used communication ports such as 443 (HTTPS) and 80 (HTTP) is essential.

SSL ensures data confidentiality and integrity as well as proving server authentication when a client establishes/accepts a connection which is mostly the case in Business-to-Customer transactions i.e., online shopping.

As verification between the server and client takes place during the SSL handshake process, important information about the certificates, security cipher, master and session keys, public or private encryption etc are determined. It is imperative that authentication take place individually both on the client and server end. The goals of SSL are *Confidentiality of communications* (primary use), *Integrity of Data* (primary use—not noticed by users), *Authentication of Server* (relies on user to be technically well informed) and *Authentication of Client* (rarely used, but has applications for SSL VPN).

Secure Electronic Transaction (SET)

Secure Electronic Transaction (SET) is an open encryption and security specification that is designed for protecting credit card transactions on the Internet. The pioneering work in this area was done in 1996 by MasterCard and Visa jointly. The need for SET came from the fact that MasterCard and Visa realized that for e-commerce payment processing, software vendors were coming up with new and conflicting standards. SET protocol testing can be performed from various perspectives i.e., from the cardholder's perspective, the payment gateway, the merchant, or perhaps even the acquirer.

Secure HTTP (S-HTTP)

HTTP is a standard Web mechanism for state management that allows a server to communicate with a client for purposes of:

- Authentication
- Security
- End user identification
- Preserving application state

S-HTTP is a protocol to transfer encrypted data over the Internet. S-HTTP testing aims to ensure that all relevant aspects of e-commerce hosting (primary use of S-HTTP) are reliable and cannot be compromised.

For example, use of dedicated IP address to enable verification of the security certificate. Testing may also encompass to see if all linked URLs include the full server path i.e., <https://>. For images, relative paths to the secure server would need to be used to avoid unexpected prompts to be displayed like "Insecure data found. Continue?" Testing will also check to see if the forms that request and collect data are secured, rather than the entire website which can slow down the customer's browsing experience.

Secure Shell (SSH)

Secure Shell Testing is applicable where SSH is used to log into another computer over a network. It's basically a network protocol that provides secure communication between two computers. If SSH is used correctly, no eavesdropping or tampering with your data is possible, unless you are under attack by an immortal miscreant with extraordinarily powerful computers. Because it proves strong authentication and secure communication, it is more reliable than telnet, FTP etc.

SSH is available in many forms -- as proprietary, open source, and free software from a variety of organizations. The most popular implementation is OpenSSH and this is the one that is part of Mac OS X.

Since a variety of authentication and encryption ciphers are used by SSH, the exact nature of testing will depend on the combination of authentication and encryption in place.

Typically, SSH is used to securely log in to remote machines in order to execute commands. You can also use this secure connection to transfer files, run GUI programs under X windows, and forward ports. It is this last feature that will enable us to create secure tunnels between computers.

The purpose of an SSH tunnel is to secure the "last mile" of your connection. There are many reasons to want this, but the most common are in situations where your connection is monitored, restricted, or not secure. For example, if you are at a coffee shop using your laptop over the provided unencrypted WiFi connection, there is a chance that someone is eavesdropping on your communication. In such a situation, you could create a tunnel to a home machine to check your email securely.

Also, many people have certain kinds of applications or sites blocked or monitored by their place of work (or by their country). Setting up an SSH tunnel to a machine outside your work (or country) network will allow you to keep your communication private and to use applications that may otherwise be blocked.

To set up an SSH tunnel, you require two computers -- one as the SSH client (machine C) and another as the SSH server (machine S). Machine C is insecure or restricted, while Machine S has full access to the Internet and is preferably under your control. In the example of an insecure wireless computer, Machine C is your wireless laptop and Machine S is a desktop connected to the Internet. In the example of securing your work communication, Machine C would be your work computer and Machine S could be a home computer connected to the Internet.

IPSec

IPSec protocol is popularly used to secure end to end communication, over public and private networks. Because it uses encryption and cryptography, it enables security to be easily customized. Similar to SSH, depending on encryption and algorithms used, testing will aim to uncover holes and weaknesses in the particular implementation in place i.e., modes of operation like transport and tunnel.

Verifying Security

Security testing aims to address inherent flaws and weaknesses in applications to ensure that the final product is robust, and can safeguard sensitive data without compromising data confidentiality and integrity. Depending on testing time frames, more specific, or thoroughly comprehensive testing can be performed to determine the robustness of an application in the environment within which it will eventually be deployed.

The goal is to verify that the application properly encrypts all authenticated and sensitive communications.

Automated approaches: Vulnerability scanning tools can verify that SSL is used on the front end, and can find many SSL related flaws. However, these tools do not have access to backend connections and cannot verify that they are secure. Static analysis tools may be able to help with analyzing some calls to backend systems, but probably will not understand the custom logic required for all types of systems. The most important protection is to use SSL on any authenticated connection or whenever sensitive data is being transmitted. There are a number of details involved with configuring SSL for web applications properly, so understanding and analyzing your environment is important. For example, IE 7.0 provides a green bar for high trust SSL certificates, but this is not a suitable control to prove safe use of cryptography alone.

Manual approaches: Testing can verify that SSL is used and find many SSL related flaws on the front end, but the automated approaches are probably more efficient. Code review is quite efficient for verifying the proper use of SSL for all backend connections.

Few Considerations for securing your communication is as under:

- Use SSL for all connections that are authenticated or transmitting sensitive or value data, such as credentials, credit card details, health and other private information.
- Ensure that communications between infrastructure elements, such as between web servers and database systems, are appropriately protected via the use of transport layer security or protocol level encryption for credentials and intrinsic value data.

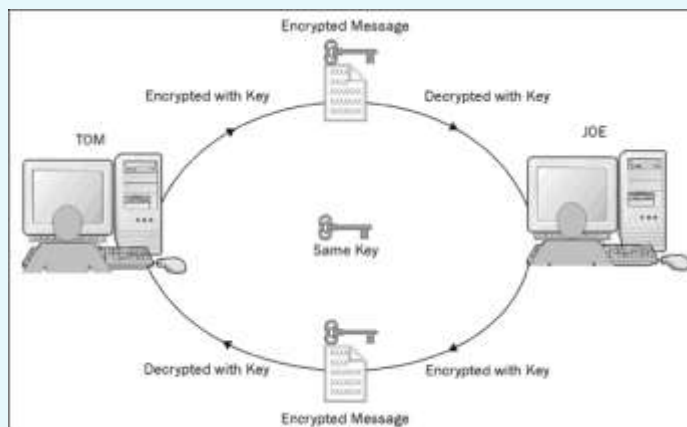


Figure 2

- Under PCI Data Security Standard requirement 4, you must protect cardholder data in transit. PCI DSS compliance is mandatory by 2008 for merchants and anyone else dealing with credit cards. In general, client, partner, staff and administrative online access to systems must be encrypted using SSL or similar. For more information, please see the PCI DSS Guidelines and implement controls as necessary.

ENDING NOTES

Security testing has gained unparalleled significance over the years, and will continue to rise up the popularity chart given the frequent advancements that we have come to notice and appreciate in the world of technology. Moreover, the importance of data integrity and confidentiality have begun to play a significant role, as have the client demands increased in terms of having access to secure software which delivers without compromising sensitive information. Our aim is to perform thorough security testing to highlight shortcomings and weaknesses in applications to enable developers to build more reliable, and dependable applications in future.

To prevent insecure communications from occurring, the first step is to make sure the security architect has formulated secure methods of communication between the clients and servers. The security architect can limit the connections they need to look at by only reviewing which servers and clients pass sensitive data.

Keep in mind, most of these architectures will fail to forget to encrypt data on back-end connections, such as database connections. Just because the data is now behind a firewall doesn't mean it should be passed in clear-text.

REFERENCES

1. <http://www.isaca.org/Journal/Past-Issues/2000/Volume-1/Pages/Secure-E-Business.aspx>
2. http://sslvpnbook.packtpub.com/preview_chapter3.htm