OWASP Application Security FAQ

From OWASP

Login Issues

What are the best practices I should remember while designing the login pages?

- From the login page, the user should be sent to a page for authentication. Once authenticated, the user should be sent to the next page. This is explained in the answer to the next question.
- The password should never be sent in clear text (unencrypted) because it can be stolen by sniffing; saving the password in clear text in the database is dangerous too.
- The best way to manage sessions would be to use one session token with two values during authentication. One value before authentication and one after.

Is it really required to redirect the user to a new page after login?

Is it really required to redirect the user to a new page after login?

Yes. Consider the application has a login page that sends the username and password as a POST request to the server. If a user clicks refresh on the second page (the page after login), the same request including the username and password in the POST will be sent again. Now suppose a valid user browses through our application and logs out, but does not close the window. The attackers come along and click the back button of the browser till they reach the second page. They only have to do a refresh and since the username and password are resubmitted and revalidated, the attackers can login as the user. Now let's assume the application has a login page which takes the user to an intermediate page for authentication. Once authenticated, the user is redirected to the second page with a session token. In this case, even if the attackers reach the second page and do a refresh, the username and password will not be resubmitted. This is so because the request that will be submitted is the one for the second page which does not contain the username and password. Therefore, it is always better to redirect the user.

How can my "Forgot Password" feature be exploited?

The Forgot Password feature is implemented in a number of different ways. One common way is to ask the user a hint question for which the user has submitted the answer during registration. These are questions like What is your favorite color? or What is your favorite pastime? If the answer is correct, either the original password is displayed or a temporary password is displayed which can be used to log in. In this method, an attacker trying to steal the password of a user may be able to guess the correct answer of the hint question and even reset the password.

In "Forgot Password", is it safe to display the old password?

If the old password is displayed on the screen, it can be seen by shoulder surfers. So it is a good idea not to display the password and let the user change to a new one. Moreover, displaying the password means it has to be stored in a recoverable form in the database which is not a good practice. If the password is stored as a one way hash in the database, the only way Forgot Password can be implemented is by letting the user reset the old password. So, it is always better to force the users reset their passwords when they forget their passwords. (A one way hash is the result obtained when we pass a string to a one way hash function. The result is such that it is impossible to get back the original value from it. Passwords are best stored as non-recoverable hashes in the database.)

Is there any risk in emailing the new password to the user's authorized mail id?

Emailing the actual password in clear text can be risky as an attacker can obtain it by sniffing. Also the mail containing the password might have a long life time and could be viewed by an attacker while it is lying in the mailbox of the user.

Apart form the above threats, a malicious user can do shoulder-surfing to view the password or login credentials.

What is the most secure way to design the Forgot Password feature?

We should first ask the user to supply some details like personal details or ask a hint question. Then we should send a mail to the users authorized mail id with a link which will take the user to a page for resetting the password. This link should be active for only a short time, and should be SSL- enabled. This way the actual password is never seen. The security benefits of this method are: the password is not sent in the mail; since the link is active for a short time, there is no harm even if the mail remains in the mailbox for a long time.

How do I protect against automated password guessing attacks?

Password guessing with automated tools is a serious problem since there are a number of tools available for this purpose. These tools essentially keep trying out different passwords till one matches. Locking out the account after 5 failed attempts is a good defense against these tools. However, the important point then is how long you lock out the account for. If it is for too long, service to valid users might be denied as the attackers repeatedly lock out your users. If the time is too short say about 1-2 minutes, the tool could start again after the timeout. So the best method would be to insist on human intervention after a few failed attempts. A method used by a number of sites these days is to have the user read and enter a random word that appears in an image on the page. Since this cannot be done by a tool, we can thwart automated password guessing. The following are some tools that guess passwords of web applications: Brutus - http://www.hoobie.net/brutus/ WebCracker http://www.securityfocus.com/tools/706

How can I protect against keystroke loggers on the client machine?

Keystroke loggers on the end users machines can sometimes ruin all our efforts of securely transmitting and storing the passwords. The users themselves may not be aware that a key logger has been installed on their machines and records each key pressed. Since the highest risk is with the password, if we can authenticate the users without having them use the keyboard, or reveal the entire password, we solve the problem. The different ways of doing this are:

- Having a graphical keyboard where the users can enter the characters they want by clicking the mouse on it. This is especially useful for numeric PINs.
- Asking the users to type a part of their password each time and not the whole password. For example you could say "Please enter the 1st, 3rd and 6th letters of your password" and this rule could be a random one each time.

My site will be used from publicly shared computers. What precautions must I take?

If the application will be accessed from publicly shared computers such as libraries, the following may protect its security.

- Avoid caching the site's pages on the system by setting the correct cache control directives.
- Exclude sensitive information from the site's URLs since the history of the client browser will store these.
- Consider protecting user input against external recording via keyboard loggers or video cameras. A graphical keyboard or prompts for a varying part of the password may help.
- Use TLS to prevent sniffing or modifying sensitive data in transit.
- Avoid weak hash algorithms in storing sensitive data by making it harder to inverse the hash. The clear text password in the memory should be reset after computing the hash.

SQL Injection

What is SQL Injection?

SQL Injection is a technique by which attackers can execute SQL statements of their choice on the backend database by manipulating the input to the application. Let's understand SQL Injection through the example of a login page in a web application where the database is SQL Server. The user needs to input Username and Password in the text boxes in Login.asp page. Suppose the user enters the following: Username : Obelix and Password : Dogmatix This input is then used to build a query dynamically which would be something like: SELECT * FROM Users WHERE username= 'Obelix' and password='Dogmatix' This query would return to the application a row from the database with the given values. The user is considered authenticated if the database returns one or more rows to the application. Now, suppose an attacker enters the following input in the login page: Username : ' or 1=1–– The query built will look like this: SELECT * FROM Users WHERE username= *or 1=1–– and password=* –– in SQL Server is used to comment out the rest of the line. So, our query is now effectively: SELECT * FROM Users WHERE username= *or 1=1 This query will look in the database for a row where either username is blank or the condition 1=1 is met. Since the latter always evaluates to true, the query will return all rows of the Users table and the user is authenticated. The attacker has been successful in logging into the application without a username and password. You can read more on this at the Securiteam site: http://www.securiteam.com/securityreviews/5DP0N1P76E.html*

Is it just ASP and SQL Server or are all platforms vulnerable?

Almost all platforms are vulnerable to SQL Injection. Inadequate checking of user input and the use of dynamic SQL queries are what make an application vulnerable to these attacks. The syntax of the input entered for SQL Injection will depend on the database being used. During our application security audits we have found many applications using other databases to be vulnerable. The above example would work on SQL Server, Oracle and MySQL. This shows that the problem is with the inadequate checking of user input and the use of dynamic SQL and not the underlying database.

Apart from username and password which variables are candidates for SQL Injection?

Any input field that makes up the where clause of a database query is a candidate for SQL Injection, eg. account numbers, and credit card numbers in the case of an online banking application. In addition to form fields, an attacker can use hidden fields and query strings also for injecting commands.

Apart from input fields, URL parameters are also vulnerable to SQL Injection as well as other input based attacks.

How do we prevent SQL Injection in our applications?

It is quite simple to prevent SQL injection while developing the application. You need to check all input coming from the client before building a SQL query. The best method is to remove all unwanted input and accept only expected input. While server side input validation is the most effective method of preventing SQL Injection, the other method of prevention is not using dynamic SQL queries. This can be achieved by using stored procedures or bind variables in databases that support these features. For applications written in Java, CallableStatements and PreparedStatements can be used. For ASP applications, ADO Command Objects can be used. You can check the following article for more on SQL Injection in Oracle: http://www.integrigy.com/info/IntegrigyIntrotoSQLInjectionAttacks.pdf

I'm using stored procedures for authentication, am I vulnerable?

Maybe, but probably no. Using stored procedures can prevent SQL Injection because the user input is no longer used to build the query dynamically. Since a stored procedure is a group of precompiled SQL statements and the procedure accepts input as parameters, a dynamic query is avoided. Although input is put into the precompiled query as is, since the query itself is in a different format, it does not have the effect of changing the query as expected. By using stored procedures we are letting the database handle the execution of the query instead of asking it to execute a query we have built. The exception to this is where the stored procedure takes a string as input and uses this string to build the query without validating it. While this is more difficult to exploit, this scenario still often leads to successful SQL Injection. This article explains how SQL Injection affects stored procedures in more detail: http://palisade.plynt.com/issues/2006Jun/injection-stored-procedures/

I'm using client side JavaScript code for checking user input. Isn't that enough?

No. Although client side checking disallows the attacker to enter malicious data directly into the input fields, that alone is not enough to prevent SQL Injection. Client side scripts only check for input in the browser. But this does not guarantee that the information will remain the same till it reaches the server. To bypass client side JavaScript, the attacker can trap the request in a proxy (eg. WebScarab, Paros (http://www.parosproxy.org)) after it leaves the browser and before it reaches the server and there he can alter input fields. The attacker can also inject commands into the querystring variables which are not checked by the client side scripts, or could disable JavaScript rendering client-side scripting useless.

Are Java servlets vulnerable to SQL injection?

Yes, they are if the user input is not checked properly, and if they build SQL queries dynamically. But Java servlets also have certain features that prevent SQL Injection like CallableStatements and PreparedStatements. Like stored procedures and bind variables, they avoid the need of dynamic SQL statements.

Can an automated scanner discover SQL Injection?

Sometimes yes, sometimes no. Whether a scanner can discover SQL injection or not depends on a variety of factors: the discovery technique used, the response from the application when a malformed SQL snippet is added, and some luck. Specifically, scanners that use Blind SQL Injection are most likely to detect SQL Injection. Scanners that claim hundreds of test cases for SQL Injection are misleading. This entry from the Penetration Testing Learning Center (http://www.plynt.com/resources/learn/tools/do_scanners_catch_sql_injectio_1/) explains this in detail.

Variable Manipulation

Why can't I trust the information coming from the browser?

There are chances that the information is modified before it reaches the server. Attackers browsing the site can manipulate the information in a GET or POST request. There are a number of HTTP/HTTPS proxy tools like Achilles (http://www.mavensecurity.com/achilles), Paros, WebScarab, etc which are capable of intercepting all this information and allow the attacker running the tool to modify it. Also, the information that the user sees or provides on a web page has to travel through the internet before it reaches the server. Although the client and the server may be trusted, we cannot be sure that the information is not modified after it leaves the browser. Attackers can capture the information on the way and manipulate it.

What information can be manipulated by the attacker?

Manipulating the variables in the URL is simple. But attackers can also manipulate almost all information going from the client to the server like form fields, hidden fields, content-length, session-id and http methods.

How do attackers manipulate the information? What tools do they use?

For manipulating any information, including form fields, hidden variables and cookies, attackers use tools known as HTTP/HTTPS proxy tools. Once the browser's proxy settings are configured to go through the HTTP/HTTPS proxy, the tool can see all information flowing between the client and the server; it even allows the attacker to modify any part of the request before sending it. Some such tools are: WebScarab can be downloaded from the OWASP site. Odysseus can be found at http://www.bindshell.net/tools/odysseus Paros can be downloaded from http://www.parosproxy.org

I'm using SSL. Can attackers still modify information?

Although SSL provides a lot of security, SSL alone is not enough to prevent variable manipulation attacks. SSL was supposed to prevent against Man in the Middle attacks but it is vulnerable to it. To successfully carry out the MITM attack, first the attacker has to divert the victim's requests to his machine i.e. redirecting the packets meant for the server to himself. He can do this by ARP poisoning / DNS Cache poisoning. Once he is able to redirect, he can see all the requests the victim is trying to make. Now when the victim tries to establish an SSL connection with a legitimate server, he gets connected to the attacker. The attacker, during the SSL Handshaking, provides a fake certificate to the victim, which the victim accepts even though the browser warns him. Thus, the victim establishes an SSL connection with the attacker instead of the server. The attacker establishes a different SSL connection with that legitimate server, which the victim was trying to connect. Now all data flow between the victim and the server will be routed through the attacker and the attacker can see all data the victim (as well as the server) sends. This is because the victim will encrypt all data with the attacker's public key, which the attacker can decrypt with his private key. The attacker can then manipulate all data that is passing through his machine.

Is there some way to prevent these proxy tools from editing the data?

The main threat these proxy tools pose is editing the information sent from the client to the server. One way to prevent it is to sign the message sent from the client with a Java Applet downloaded onto the client machine. Since the applet we developed will be the one validating the certificate and not the browser, a proxy tool will not be able to get in between the client and the server with a fake certificate. The applet will reject the fake certificate. The public key of this certificate can then be used to digitally sign each message sent between the client and the server. An attacker would then have to replace the embedded certificate in the applet with a fake certificate to succeed – that raises the barrier for the attacker.

Browser Cache

How can the browser cache be used in attacks?

The browser has a capability to temporarily store some of the pages browsed. These cached files are stored in a folder, like the Temporary Internet Files folder in the case of Internet Explorer. When we ask for these pages again, the browser displays them from its cache. This is much faster than downloading the page from the server. Let's consider the particular scenario where a user has logged in to an application with username and password. The user browses the different pages which contain sensitive information. Let's suppose a page with the user's credit card information gets cached in the browser and the user logs out of the application. Now suppose the attackers access the same machine and searches through the Temporary Internet Files, they will get the credit card details. The attackers do not need to know the username and password of the user to steal the information.

How do I ensure that sensitive pages are not cached on the user's browser?

The response header sent from the server has some cache control directives that can be set from your code. These directives control the caching of content on any cache. The directives to be set are Cache-Control : no-cache, no-store and Expires: 0. But since legacy HTTP 1.0 servers do not support the Cache-Control headers, universally, Pragma: no-cache header should be used, too.

What's the difference between the cache-control directives: no-cache, and no-store?

The no-cache directive in a response indicates that the response must not be used to serve a subsequent request i.e. the cache must not display a response that has this directive set in the header but must let the server serve the request. The no-cache directive can include some field names; in which case the response can be shown from the cache except for the field names specified which should be served from the server. The no-store directive applies to the entire message and indicates that the cache must not store any part of the response or any request that asked for it.

Am I totally safe with these directives?

No. But generally, use both Cache-Control: no-cache, no-store and Pragma: no-cache, in addition to Expires: 0 (or a sufficiently backdated GMT date such as the UNIX epoch). Non-html content types like pdf, word documents, excel spreadsheets, etc often get cached even when the above cache control directives are set (although this varies by version and additional use of must-revalidate, pre-check=0, post-check=0, max-age=0, and s-maxage=0 in practice can sometimes result at least in file deletion upon browser closure in some cases due to browser quirks and HTTP implementations). Also, 'Autocomplete' feature allows a browser to cache whatever the user types in an input field of a form. To check this, the form tag or the individual input tags should include 'Autocomplete="Off" ' attribute. However, it should be noted that this attribute is non-standard (although it is supported by the major browsers) so it will break XHTML validation.

Where can I learn more about caching?

Some useful links that talk about caching are - Caching Tutorial for Web Authors and Webmasters by Mark Nottingham at http://www.mnot.net/cache_docs/ and HTTP RFC (sec14.9.1) at http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html

Cross Site Scripting

What is Cross Site Scripting?

Cross Site scripting (XSS) is a type of attack that can be carried out to steal sensitive information belonging to the users of a web site. This relies on the server reflecting back user input without checking for embedded javascript. This can be used to steal cookies and session IDs. Let's see how it works. We would all have come across the following situation sometime – we type a URL in the browser, say www.abcd.com/mypage.asp, and receive an error page that says "Sorry www.abcd.com/mypage.asp does not exist" or a page with a similar message. In other words, pages that display the user input back on the browser. Pages like this could be exploited using XSS. Instead of a normal input, think what will happen if the input contains a script in it. While reflecting back the input, instead of rendering it as normal HTML output, the browser treats it as a script and executes it. This script could contain some malicious code. The attackers can send a link that contains a script as part of the URL to a user. When the user clicks it, the script gets executed on the user's browser. This script may have been written to collect important information about the user and send it to the attacker. Kevin Spett's paper Cross Site Scripting, Are your web applications vulnerable? is a good source of information on this topic and is available at http://www.spidynamics.com/whitepapers/SPIcross-sitescripting.pdf The Cross Site Scripting FAQ at CGI Security is another good place to learn more on XSS.

What information can an attacker steal using XSS?

The attackers can steal the session ID of a valid user using XSS. The session ID is very valuable because it is the secret token that the user presents after login as proof of identity until logout. If the session ID is stored in a cookie, the attackers can write a script which will run on the user's browser, query the value in the cookie and send it to the attackers. The attackers can then use the valid session ID to browse the site without logging in. The script could also collect other information from the page, including the entire contents of the page.

Apart from mailing links of error pages, are there other methods of exploiting XSS?

Yes, there are other methods. Let's take the example of a bulletin board application that has a page where data entered by one user can be viewed by other users. The attackers enter a script into this page. When a valid user tries to view the page, the script gets executed on the user's browser. It will send the user's information to the attackers.

How can I prevent XSS?

XSS can be prevented while coding the application. You should be validating all input and output to and from the application and escape all special characters that may be used in a script. If the code replaces the special characters by the following before displaying the output, XSS can be prevented to some extent.

Special Character	Escape Sequence
<	&lt;
>	&gt;
(	&#40;
)	&#41;
*	&#42;
&	&amp;

Gunter Ollmann has written an excellent paper on the use of special characters in XSS attacks. For instance, the above technique of escaping special characters cannot protect against a script injected like "javascript:self.location.href = 'www.evil.org' " as this script does not use any of the special characters.

Can XSS be prevented without modifying the source code?

There is a method that requires minimal coding as compared to performing input, output validation to prevent the stealing of cookies by XSS. Internet Explorer 6 has an attribute called HTTP Only that can be set for cookies. Using this attribute makes sure that the cookie can not be accessed by any scripts. More details are available at the MSDN site on httpcookies at http://msdn.microsoft.com/library/default.asp?url=/workshop/author/dhtml/httponly_cookies.asp Mozilla also has plans to implement a similar feature. Researchers have found a method to beat this. It is known as Cross Site Tracing.

What is Cross Site Tracing (XST)? How can it be prevented?

Attackers are able to bypass the HTTP Only attribute to steal cookie information by Cross Site tracing (XST). TRACE is a HTTP method that can be sent to the server. The server sends back anything included in the TRACE request back to the browser. In a site that uses cookies, the cookie information is sent to the server in each request. If we send a TRACE request in a URL of such a site, the server will send back all cookie information to the browser. Now imagine a situation similar to the one described in XSS but the site in this case is using the HTTP Only cookies. The attackers make a valid user click on a link that contains a script that calls the TRACE method. When the user clicks on the link the TRACE request as well as all the cookie information is sent to the server. The server then sends back the cookie information back to the script in the browser. Suppose that the malicious script also contains code to send this information to the attackers. The attackers have succeeded again in stealing the cookies although HTTP Only Cookies were used. To summarize, HTTP Only cookies prevent the JavaScript from directly accessing the cookies but the attacker was able to retrieve it through an indirect method. XST can be prevented by disabling the TRACE method on the web server. This paper by Jeremiah Grossman discusses XST in greater detail http://www.cgisecurity.com/whitehat-mirror/WhitePaper_screen.pdf

Web Server Fingerprinting

How do attackers identify which web server I'm using?

Identifying the application running on a remote web server is known as fingerprinting the server. The simplest way to do this is to send a request to the server and see the banner sent in the response. Banners will generally have the server name and the version number in it. We can address this problem by either configuring the server not to display the banner at all or by changing it to make the server look like something else.

How can I fake the banners or rewrite the headers from my web server?

There are a number of tools that help in faking the banners. URLScan is a tool that can change the banner of an IIS web server. http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/tools/URLScan.asp mod_security has a feature for changing the identity of the Apache web server. It can be found at http://www.modsecurity.org/ Servermask for faking banners of IIS, can be found at http://www.servermask.com/

Once I fake the banners, can my web server still be fingerprinted?

Yes. Unfortunately there are tools that fingerprint the web server without relying on the banners. Different web servers may implement features not specified in HTTP RFCs differently. Suppose we make a database of these special requests and the responses of each web server. We can now send these requests to the web server we want to fingerprint and compare the responses with the database. This is the technique used by tools like Fire & Water. This tool can be found at http://www.ntobjectives.com/products/firewater/ There is a paper by Saumil Shah that discusses the tool httprint at http://net-square.com/httprint/httprint_paper.html httprint can be found at http://net-square.com/httprint/

A friend told me it's safer to run my web server on a non-standard port. Is that right?

A web server generally needs to be accessed by a lot of people on the internet. Since it normally runs on port 80 and all browsers are configured to access port 80 of the web server, users are able to browse the site. If we change the port, the users will have to specify the port in addition to the domain name. But this is a good idea for an intranet application where all users know where to connect. It is more secure since the web server will not be targeted by automated attacks like worms that scan port 80 and other standard ports.

Should I really be concerned that my web server can be fingerprinted?

Well, there are two schools of thought here. According to the first school, yes you should take precaution against fingerprinting as correctly identiying the web server maybe the first step in a more dangerous attack. Once attackers have found out that the web server is say IIS 5, they will search for known vulnerabilities for IIS 5. If the web server is not patched for all known vulnerabilities or the attackers find one for which a patch has not been released yet, there is nothing to stop them from attacking it. Also automated tools and worms can be fooled by changing the version information. Some determined and focused attackers might go to additional lengths to identify the server but the hurdles that the attackers have to overcome have increased when it's more difficult to fingerprint the web server name and version. Jeremiah Grossman pointed out the other school of thought. Evasive measures are futile as any scanner targeting a web site, will normally not care what the web server is. The scanner will run ALL its tests no matter if they apply to the system or not. This is a typical shotgun approach. A bad guy targeting the site might be hampered by not knowing the exact version, but if he's determined he would still try out all related exploits and try to break in.

Testing

I want to chain my proxy tool with a proxy server; are there tools that let me do that?

Yes, there are several tools that allow proxy chaining. Some of these are: WebScarab – http://www.owasp.org/development/webscarab Exodus – http://home.intekom.com/rdawes/exodus.html Odysseus – http://www.wastelands.gen.nz/odysseus/index.php

Can't web application testing be automated? Are there any tools for that?

There are tools that scan applications for security flaws. But these tools can only look for a limited number of vulnerabilities, and do not find all the problems in the application. Moreover, a lot of attacks require understanding of the business context of the application to decide on the variables to manipulate in a particular request, which a tool is incapable of doing. A presentation by Jeremiah Grossman of White Hat Security which talks about the limitations of automated scanning (http://www.blackhat.com/presentations/bh-federal-03/bh-fed-03-grossman-up.pdf).

This piece explains what a scanner can't find (http://www.plynt.com/resources/learn/tools/what_cant_a_scanner_find_1/).

In our tests using a slightly modified WebGoat the best Black-box scanning tool found less than 20% of the issues ! Some tools for automated scanning are: SpikeProxy, open source and freely available at http://www.immunitysec.com/spikeproxy.html WebInspect, can be found at http://www.spidynamics.com/productline/WE_over.html

Where can I try out my testing skills? Is there a sample application I can practice with?

OWASP provides a sample application that can be used for this purpose called . As the site says, the WebGoat project's goal is to teach web security in an interactive teaching environment. There are lessons on most of the common vulnerabilities. Another interesting site is Hackingzone which has a game on SQL Injection at http://www.hackingzone.org/sql/index.php

Are there source code scanning tools for .NET languages, Java, PHP etc that predict vulnerabilities in the source code?

Rough Auditing Tool for Security (RATS) is a tool that scans the source code for security flaws in C, C++, Python, Perl and PHP programs. It can be found at http://code.google.com/p/rough-auditing-tool-for-security/downloads/list FX Cop was created by the Microsoft Team at the GotDotNet community site to check for the .NET Frameowork guidelines which include security. Prexis is a commercial source code and run-time analyzer. Flawfinder is a static source code analyzer. Compaq ESC is a run-time analyzer for Java. Parasoft AEP is a commercial source code analyzer for Java. Fortify SCA from Fortify Software is another source code analyzer that supports mixed language analysis of C, C++, C#, ASP.NET, Java, JSP, PL/SQL, VB.NET, XML, etc. Secure Coding plugins are also available. Similar source code analyzers are Klocwork K7 for C, C++ and Java; Coverity Prevent for detecting security violations and defects in code; Ounce Solutions for C, C++, C#, ASP.NET, Java and JSP. We would like to know about more tools for scanning source code. If you know about any, please inform us and we'll add to this FAQ

Can non-HTTP protocols also be intercepted and played with like this?

Yes, Interactive TCP Replay is a tool that acts as a proxy for non-HTTP applications and also allows modifying the traffic. It allows editing of the messages in a hex editor. ITR also logs all the messages passing between the client and the server. It can use different types of character encoding like ASCII or EBCDIC for editing and logging. More information on this can be found at http://www.webcohort.com/web_application_security/research/tools.html

Cryptography/SSL

What is SSL?

Secure Socket Layer (SSL) gives us assurance of two things. Firstly when a client connects to a web server, the client can be sure that it is talking to the right server by checking the certificate the server sends it. Secondly, SSL assures you of the confidentiality of the data, as the client and the server exchange encrypted messages that cannot be understood by anybody else. This is how SSL works: When the client requests for a SSL page, the server sends a certificate that it has obtained from a trusted certificate authority. This certificate contains the public key of the server. After satisfying itself that the certificate is correct and the server is a genuine one, the client generates one random number, the session key. This key is encrypted by the public key of the server and sent across. The server decrypts the message with its private key. Now both sides have a session key known only to the two of them. All communication to and fro is encrypted and decrypted with the session key. An interesting link on SSL is http://www.rsasecurity.com/standards/ssl/basics.html

Should I use 40-bit or 128-bit SSL?

There are 2 strengths in SSL - 40-bit and 128-bit. These refer to the length of the secret key used for encrypting the session. This key is generated for every SSL session and is used to encrypt the rest of the session. The longer the key the more difficult it is to break the encrypted data. So, 128-bit encryption is much more secure than 40-bit. Most browsers today support 128-bit encryption. There are a few countries which have browsers with only 40-bit support. In case you are using 40-bit SSL, you may need to take further precautions to protect sensitive data. Salted hash for transmitting passwords is a good technique. This ensures that the password can not be stolen even if the SSL key is broken.

Is 40-bit SSL really unsafe?

40-bit SSL is not really unsafe. It's just that it is computationally feasible to break the key used in 40-bit but not the key used in 128-bit. Even though 40-bit can be broken, it takes a fairly large number of computers to break it. Nobody would even attempt to do that for a credit card number or the like. But there are claims of breaking the 40-bit RC4 key in a few hours. So depending on the data your application deals with, you can decide on the SSL strength. Using 128-bit is definitely safer.

With home computers gtting faster day by day, a dedicated, expensive and very fast computer can break 40-bit encryption in few minutes (ideally testing a million keys per second). On the other hand, 128-bit encryotion will have about 339,000,000,000,000,000,000,000,000,000,000,000 (Couple of Trillions or 2^128) possible key combinations and it will take around 1000 Years to break 128-bit encryptions with the help of a cluster of very fast computers.

What all are encrypted when I use SSL? Is the page request also encrypted?

After the initial SSL negotiation is done and the connection is on HTTPS, everything is encrypted including the page request. So any data sent in the query string will also be encrypted.

Which cryptographic algorithms do SSL use?

SSL supports a number of cryptographic algorithms. During the initial "handshaking" phase, it uses the RSA public key algorithm. For encrypting the data with the session key the following algorithms are used - RC2, RC4, IDEA, DES, triple-DES and MD5 message digest algorithm.

I want to use SSL. Where do I begin?

There are several Certificate Authorities that you can buy a SSL certificate from. Whichever CA you choose, the basic procedure will be as follows -

- Create key pair for the server
- Create the Certificate Signing Request. This will require you to provide certain details like location and fully qualified domain name of the server.
- Submit the CSR to the CA along with documentary proof of identity.
- Install the certificate sent by the CA

The first two steps are done from the web server. All servers have these features. While installing the certificate issued by the CA, you will have to specify which web pages are to be on SSL.

A good starting point for working on POC in a Windows development environment could be: "HOW TO: Secure XML Web Services with Secure Socket Layer in Windows 2000" - http://support.microsoft.com/default.aspx?scid=kb;en-us;q307267&sd=tech

Cookies and Session Management

Are there any risks in using persistent vs non-persistent cookies?

Persistent cookies are data that a web site places on the user's hard drive (or equivalent) for maintaining information over more than one browser session. This data will stay in the user's system and can be accessed by the site the next time the user browses the site. Non-persistent cookies on the other hand are those that are used only in the browser session that creates it. They stay only in the memory of the machine and are not persisted on the hard disk. The security risk with persistent cookies is that they are generally stored in a text file on the client and an attacker with access to the victim's machine can steal this information.

Can another web site steal the cookies that my site places on a user's machine?

No, it is not possible for a website to access another site's cookies. Cookies have a domain attribute associated with them. Only a request coming from the domain specified in the attribute can access the cookie. This attribute can have only one value.

Which is the best way to transmit session ids– in cookies, or URL or a hidden variable?

Transmitting session IDs in the URL can lead to several risks. Shoulder surfers can see the session ID; if the URL gets cached on the client system, the session ID will also be stored; the session ID will get stored in the referrer logs of other sites. Hidden variables are not always practical as every request might not be a POST. Cookies are the safest method as cookies do not get cached, are not visible in the W3C or referrer logs, and most users anyway accept cookies.

What are these secure cookies?

A cookie can be marked as "secure" which ensures the cookie is used only over SSL sessions. If "secure" is not specified, the cookie will be sent unencrypted over non–SSL channels. Sensitive cookies like session tokens should be marked as secure if all pages in the web site requiring session tokens are SSL–enabled. One thing to keep in mind here is that images are generally not downloaded over SSL and they usually don't require a session token to be presented. By setting the session cookie to be secure, we ensure that the browser does not send the cookie while downloading the image over the non–SSL connection.<>

If I use a session ID that is a function of the client's IP address, will session hijacking be prevented?

An attacker can hijack another user's session by stealing the session token. Methods have been suggested to prevent the session from being hijacked even if the session token is stolen. For instance, using a session token that is a function of the user's IP address. In this approach, even if the attacker stole the token, he would need the same IP address as the user to successfully hijack a session. However, session hijacking can still be possible. Suppose the attacker is on the same LAN as the user and uses the same Proxy IP as the user to access the web site. The attacker can still steal the session if he is able to sniff the session token. It may also be not possible to implement this if the IP of the client changes during a session, making the session invalid if the token is tied to the initial IP address. This may happen if the client is coming from behind a bank of proxy servers.

How about encrypting the session id cookies instead of using SSL?

Encrypting just the session ID over a non-SSL connection will not serve any purpose. Since the session ID will be encrypted once and the same value will be sent back and forth each time, an attacker can use the encrypted value to hijack the session.

What is the concept of using a page id, in addition to the session id?

A Session ID or token has the lifetime of a session and is tied to the logged in user. A page ID or token has a lifetime of a page and is tied to a page that is served. It is a unique token given when a page is downloaded and is presented by the user when accessing the next page. The server expects a particular value for the user to access the next page. Only if the token submitted matches what the server is expecting is the next page served. An application can use this to ensure that a user accesses pages only in the sequence determined by the application. The user cannot paste a deep URL in the browser and skip pages just because he has a session token, as the page token would not be authorized to access the deeper URL directly. Good Read: Secure your sessions with Page Tokens (http://palisade.plynt.com/issues/2005Aug/page-tokens/)

Logging and Audit Trails

What are these W3C logs?

W3C is a logging format used for Web server log files. W3C logs record access details of each request: the timestamp, source IP, page requested, the method used, http protocol version, browser type, the referrer page, the response code etc. Note that these are access logs, and so a separate record is maintained for each request. When a page with multiple gif files is downloaded, it would be recorded as multiple entries in the W3C log; so, W3C logs tend to be voluminous.

Do I need to have logging in my application even if I've W3C logs?

Yes, it's important that your application maintains "application level" logs even when W3C logging is used. As W3C logs contain records for every http request, it is difficult (and, at times impossible) to extract a higher level meaning from these logs. For instance, the W3C logs are cumbersome to identify a specific session of user and the activities that the user performed. It's better that the application keeps a trail of important activities, rather than decode it from W3C logs.

What should I log from within my application?

Keep an audit trail of activity that you might want to review while troubleshooting or conducting forensic analysis. Please note that it is inadvisable to keep sensitive business information itself in these logs, as administrators have access to these logs for troubleshooting. Activities commonly kept track of are:

- Login and logout of users
- Critical transactions (eg. fund transfer across accounts)
- Failed login attempts
- Account lockouts
- Violation of policies

The data that is logged for each of these activities usually include:

- User ID
- Time stamp
- Source IP
- Error codes, if any
- Priority

Should I encrypt my logs? Isn't that a performance hit?

Encryption is required when information has to be protected from being read by unauthorized users. Yes, encryption does take a performance hit, so if your logs do not contain sensitive information you might want to forego encryption. However, we strongly urge that you protect your logs from being tampered by using digital signatures. Digital signatures are less processor intensive than encryption and ensure that your logs are not tampered.

Can I trust the IP address of a user I see in my audit logs? Could a user be spoofing/impersonating their IP address?

A bad guy who wants to hide his actual IP address might use a service like anonymizer, or use open HTTP relays. [HTTP open relays are improperly configured web servers on the web that are used as a HTTP proxy to connect to other sites.] In such cases, the IP address you see in your log files will be those of these services or the open relay that is being used. So, the IP address you see in your log files might not always be trustworthy.

Miscellaneous

What are Buffer Overflows?

Buffer overflow vulnerability affects the web applications that require user input. The application stores the input in a buffer which is of a fixed size, as defined by the programmer. When the input that is sent to the application is more than the buffer capacity and the buffers are left unchecked, buffer overflow occurs. The severity depends on the user input. If a malicious code executes as a result of the overflow, it can even compromise the whole system. To learn more, please read the OWASP article on buffer overflows.
(http://www.owasp.org/index.php/Buffer_Overflow)

What are application firewalls? How good are they really?

Application firewalls analyze the requests at the application level. These firewalls are used for specific applications like a web server or a database server. The web application firewalls protect the web server from HTTP based attacks. They monitor the requests for attacks that involve SQL Injection, XSS, URL encoding etcetera. However, application layer firewalls cannot protect against attacks that require an understanding of the business context – this includes most attacks that rely on variable manipulation. Some application firewalls are: Netcontinuum's NC-1000 Kavado Inc.'s InterDo Teros Inc.'s Teros-100 APS

What is all this about "referrer logs", and sensitive URLs?

The HTTP header contains a field known as Referrer. For visiting a web page we may either:

- Type its URL directly into the address bar of the browser
- Click a link on some other page that brings us there
- Be redirected there by some page.

In the first case, the referrer field will be empty but in the other two cases it will contain the URL of the previous page. The URL of the first page will get stored in the web server access logs of the second page when the user reaches the second page from the first page. Now suppose, the two pages belong to different sites and the first URL contains sensitive information like a user's session ID. If the second site belongs to attackers, they can obtain this information by just going through the logs. Information in the URLs will get stored in the referrer logs as well as the history of the browser. Therefore, we should be careful not to have any sensitive information in the URL.

I want to use the most secure language; which language do you recommend?

Any language can be used since secure programming practices are what make applications safe. Most security techniques can be implemented in any language. Our advice would be to use any language you are comfortable with. But some languages like Java have additional features like bind variables that aid security; you could use those additional features if you decide to program in that language.

What are the good books to learn secure programming practices?

The OWASP Guide to Building Secure Web Application and Web Services is a good guide for web application developers. You can download it from http://www.owasp.org/documentation/guide Writing Secure Code by Michael Howard and David LeBlanc has a chapter on Securing Web-Based Services. More information on this book can be found at: http://www.microsoft.com/mspress/books/toc/5612.asp Secure Programming for Linux and Unix HOWTO by David Wheeler talks about writing secure applications including web applications; it also specifies guidance for a number of languages. The book can be found at: http://www.dwheeler.com/secure-programs

Another good book on application security, which also covers some web service specific topics: 19 Deadly Sins of Software Security, by: Michael Howard, David LeBlanc and John Viega (http://books.mcgraw–hill.com/getbook.php?isbn=0072260858).

Are there any training programs on secure programming that I can attend?

Microsoft offers training programs on Developing Security–Enhanced Web Applications and Developing and Deploying Secure Microsoft .NET Framework Application. More information can be found at http://www.microsoft.com/traincert/syllabi/2300AFinal.asp and http://www.microsoft.com/traincert/syllabi/2350BFinal.asp Foundstone offers secure coding training through Global Knowledge Aspect Security offers a similar course.

OWASP_FAQ_Ver3.doc