

Security Knowledge Framework

Function Name: foobarFunction Description: lalala  
  
Date: 2015-02-09 22:34  
  
Project: qwe

# Table of contents

Introduction

filename injection

Open forward &amp; Open redirects

File upload injections

GET/POST requests

# Introduction

The security knowledge framework is composed by means of the highest security standards currently available and is designed to maintain the integrety of your application, so you and your costumers sensitive data is protected against hackers. This document is provided with a checklist in which the programmers of your application had to run through in order to provide a secure product.

In the post-development stage of the security knowledge framework the developer double-checks his application against a checklist which consists out of several questions asking the developer about different stages of development and the methodology of implementing different types of functionality the application contains. After filling in this checklist the developer gains feedback on the failed checklist items providing him with solutions about how to solve the additional vulnerability's found in the application.

# filename injection

Description:  
  
Solution:  
The most effective solution to eliminate file inclusion vulnerabilities is to avoid passing user-submitted input to any filesystem/framework API. If this is not possible the application can maintain a white list of files, that may be included by the page, and then use an identifier (for example the index number) to access to the selected file. Any request containing an invalid identifier has to be rejected, in this way there is no attack surface for malicious users to manipulate the path.

# Open forward &amp; Open redirects

Description:  
An open forward is an application that takes a parameter and forwards a user to another part of the application without any validation or access control checks. This may allow an attacker to bypass access control checks, especially those enforced externally, such as by a web server.   
An open redirect is an application that takes a parameter and redirects a user to the parameter value without any validation. This vulnerability is used in phishing attacks to get users to visit malicious sites without realizing it.   
  
Solution:  
Safe use of redirects and forwards can be done in a number of ways:  
1.  
Simply avoid using redirects and forwards.  
2.  
If used, do not involve user parameters in calculating the destination. This can usually be done.  
3.  
If destination parameters can not be avoided, ensure that the supplied value is valid, and authorized for the user.  
It is recommended that any such destination parameters be a mapping value, rather than the actual URL or portion of the URL, and that server side code translate this mapping to the target URL.  
  
Use a whitelisting method for determining where the user should be redirected towards.  
  
Avoiding such flaws is extremely important as they are a favorite target of phishers trying to gain the users trust.

# File upload injections

Description:  
Uploaded files represent a significant risk to applications. The first step in many attacks is to get some code to the system to be attacked. Then the attack only needs to find a way to get the code executed. Using a file upload helps the attacker accomplish the first step.  
The consequences of unrestricted file upload can vary, including complete system takeover, an overloaded file system or database, forwarding attacks to back-end systems, and simple defacement. It depends on what the application does with the uploaded file and especially where it is stored.  
There are really two classes of problems here. The first is with the file metadata, like the path and file name. These are generally provided by the transport, such as HTTP multi-part encoding. This data may trick the application into overwriting a critical file or storing the file in a bad location. You must validate the metadata extremely carefully before using it.  
The other class of problem is with the file size or content. The range of problems here depends entirely on what the file is used for. See the examples below for some ideas about how files might be misused. To protect against this type of attack, you should analyze everything your application does with files and think carefully about what processing and interpreters are involved.   
  
Solution:  
Uploaded files always needs to be placed outside the document root of the webserver. Also for serving the files back there needs to be a file handler function that can select the file based on a identifier and the file will be served to the user.

# GET/POST requests

Description:  
Authors of services which use the HTTP protocol SHOULD NOT use GET based forms for the submission of sensitive data, because this will cause this data to be encoded in the Request-URI. Many existing servers, proxies, and user agents will log the request URI in some place where it might be visible to third parties. Servers can use POST-based form submission instead.  
GET parameters are also more likely to be vulnerable to XSS. Please refer to the XSS manual in the knowledgebase for more information.  
  
Solution:  
Whenever transmitting sensitive data always do this by means of the POST request.