acunetix

WEB APPLICATION SECURITY

Acunetix Website Audit

2 April, 2007

Payment Card Industry Data Security Standard (PCI)

~ compliance report ~

Payment Card Industry Data Security Standard (PCI)

compliance report

Description

This document describes the 12 Payment Card Industry (PCI) Data Security Standard (DSS) requirements."
These security requirements apply to all "system components." System components are defined as any network component, server, or application that is included in or connected to the cardholder data environment. The cardholder data environment is that part of the network that possesses cardholder data or sensitive authentication data. Adequate network segmentation, which isolates systems that store, process, or transmit cardholder data from those that do not, may reduce the scope of the cardholder data environment. Network components include but are not limited to firewalls, switches, routers, wireless access points, network appliances, and other security appliances. Server types include but are not limited to the following: web, database, authentication, mail, proxy, network time protocol (NTP), and domain name server (DNS). Applications include all purchased and custom applications, including internal and external (Internet) applications.

Disclaimer

The information provided does not constitute legal advice. The results of a vulnerability assessment will demonstrate potential vulnerabilities in your application that should be corrected in order to reduce the likelihood that your information will be compromised. As legal advice must be tailored to the specific application of each law, and laws are constantly changing, nothing provided herein should be used as a substitute for the advice of competent counsel.
This document was generated using information provided in "Payment Card Industry Data Security Standard", that can be found at http://usa.visa.com/merchants/risk_management/cisp.html.

Scan

URL	**http://bld:80/test/actions/**
Scan date	4/2/2007 10:19:43 AM
Duration	19 seconds

Compliance at a Glance

This section of the report is a summary and lists the number of alerts found according to individual compliance categories.

- Remove all unnecessary functionality (Requirement 2.2.4)
  1 alerts
- Encrypt all non-console administrative access (Requirement 2.3)
  0 alerts
- Encrypt transmission of cardholder data across open, public networks (Requirement 4)
  0 alerts
- Develop and maintain secure systems and applications (Requirement 6)
  8 alerts
- Unvalidated Input (Requirement 6.5.1)
  6 alerts
- Broken Access Control (Requirement 6.5.2)
  0 alerts
- Broken Authentication and Session Management (Requirement 6.5.3)
  5 alerts
- Cross Site Scripting (XSS) Flaws (Requirement 6.5.4)
  5 alerts
- Buffer Overflows (Requirement 6.5.5)
  0 alerts

- Injection Flaws (Requirement 6.5.6)
  0 alerts
- Improper Error Handling (Requirement 6.5.7)
  0 alerts
- Insecure Storage (Requirement 6.5.8)
  0 alerts
- Denial of Service (Requirement 6.5.9)
  0 alerts
- Insecure Configuration Management (Requirement 6.5.10)
  12 alerts

Compliance According to Categories: A Detailed Report

This section is a detailed report that explains each vulnerability found according to individual compliance categories.

(Requirement 2.2.4) Remove all unnecessary functionality

Remove all unnecessary functionality, such as scripts, drivers, features, subsystems, file systems, and unnecessary web servers.

Total number of alerts in this category: 1

Alerts in this category

Possible sensitive files

A possible sensitive file has been found. This file is not directly linked from the website. This check looks for known sensitive files like: password files, configuration files, log files, include files, statistics data, database dumps. Each of those files may help an attacker to learn more about his target.

Affected item	**/test/actions/test.html**
Affected parameter	
Variants	1

(Requirement 2.3) Encrypt all non-console administrative access

Encrypt all non-console administrative access. Use technologies such as SSH, VPN, or SSL/TLS (transport layer security) for web-based management and other non-console administrative access.

Acunetix WVS did not found any alerts in this category.

(Requirement 4) Encrypt transmission of cardholder data across open, public networks

Sensitive information must be encrypted during transmission over networks that are easy and common for a hacker to intercept, modify, and divert data while in transit.

Acunetix WVS did not found any alerts in this category.

(Requirement 6) Develop and maintain secure systems and applications

Unscrupulous individuals use security vulnerabilities to gain privileged access to systems. Many of these vulnerabilities are fixed by vendor-provided security patches. All systems must have the most recently released, appropriate software patches to protect against exploitation by employees, external hackers, and viruses.

Total number of alerts in this category: 8

Alerts in this category

Apache Mod_Rewrite Off-By-One Buffer Overflow Vulnerability

This alert has been generated using only banner information. It may be a false positive.

Apache mod_rewrite is prone to an off-by-one buffer-overflow condition. The vulnerability arising in the mod_rewrite module's ldap scheme handling allows for potential memory corruption when an attacker exploits certain rewrite rules.

Affected Apache versions:

- Apache 1.3.28 - 1.3.36 with mod_rewrite
- Apache 2.2.0 - 2.2.2 with mod_rewrite
- Apache 2.0.46 - 2.0.58 with mod_rewrite

Affected item	**Web Server**
Affected parameter	
Variants	1

PHP HTML Entity Encoder Heap Overflow Vulnerability

This alert has been generated using only banner information. It may be a false positive.

Stefan Esser reported some vulnerabilities in PHP, which can be exploited by malicious people to cause a DoS (Denial of Service) or potentially compromise a vulnerable system.The vulnerabilities are caused due to boundary errors within the "htmlentities()" and "htmlspecialchars()" functions. If a PHP application uses these functions to process user-supplied input, this can be exploited to cause a heap-based buffer overflow by passing specially crafted data to the affected application. Successful exploitation may allow execution of arbitrary code, but requires that the UTF-8 character set is selected. For a detailed explanation of the vulnerability read the referenced article.
Vendor has released PHP 5.2.0 which fixes this issue.

Affected PHP versions (up to 4.4.4/5.1.6).

Affected item	**PHP**
Affected parameter	
Variants	1

PHP version older than 4.4.1

This alert has been generated using only banner information. It may be a false positive.

Multiple vulnerabilities have been reported in PHP, which can be exploited by malicious people to conduct cross-site scripting attacks, bypass certain security restrictions, and potentially compromise a vulnerable system.

Affected PHP versions (up to 4.4.0).

Affected item	**PHP**
Affected parameter	
Variants	1

PHP Zend_Hash_Del_Key_Or_Index vulnerability

This alert has been generated using only banner information. It may be a false positive.

Stefan Esser had discovered a weakness within the depths of the implementation of hashtables in the Zend Engine. This vulnerability affects a large number of PHP applications. It creates large new holes in many popular PHP applications. Additonally many old holes that were disclosed in the past were only fixed by using the unset() statement. Many of these holes are still open if the already existing exploits are changed by adding the correct numerical keys to survive the unset (). For a detailed explanation of the vulnerability read the referenced article.

Affected PHP versions (up to 4.4.2/5.1.3).

Affected item	**PHP**
Affected parameter	
Variants	1

Unfiltered Header Injection in Apache 1.3.34/2.0.57/2.2.1

This version of Apache is vulnerable to HTML injection (including malicious Javascript code) through "Expect" header. Until not it was not classed as security vulnerability as an attacker has no way to influence the Expect header a victim will send to a target site. However, according to Amit Klein's paper: "Forging HTTP request headers with Flash" there is a working cross site scripting (XSS) attack against Apache 1.3.34, 2.0.57 and 2.2.1 (as long as the client browser is IE or Firefox, and it supports Flash 6/7+).

Affected Apache versions (up to 1.3.34/2.0.57/2.2.1).

Affected item	**Web Server**
Affected parameter	
Variants	1

Apache version older than 1.3.34

This alert has been generated using only banner information. It may be a false positive.

Two potential security issues have been fixed in Apache version 1.3.34:

- If a request contains both Transfer-Encoding and Content-Length headers, remove the Content-Length, mitigating some HTTP Request Splitting/Spoofing attacks.
- Added TraceEnable [o|off|extended] per-server directive to alter the behavior of the TRACE method.

Affected Apache versions (up to 1.3.33).

Affected item	**Web Server**
Affected parameter	
Variants	1

Apache version up to 1.3.33 htpasswd local overflow

This alert has been generated using only banner information. It may be a false positive.

A buffer overflow vulnerability exists in the htpasswd utility included with Apache. The vulnerability is due to improper bounds checking when copying user-supplied 'user' data into local buffers.

Affected Apache versions (up to 1.3.33).

Affected item	**Web Server**
Affected parameter	
Variants	1

TRACE Method Enabled

HTTP TRACE method is enabled on this web server. In the presence of other cross-domain vulnerabilities in web browsers, sensitive header information could be read from any domains that support the HTTP TRACE method.

Affected item	**Web Server**
Affected parameter	
Variants	1

(Requirement 6.5.1) Unvalidated Input

Web applications use input from HTTP requests (and occasionally files) to determine how to respond. Attackers can tamper with any part of an HTTP request, including the url, querystring, headers, cookies, form fields, and hidden fields, to try to bypass the site's security mechanisms. Common names for common input tampering attacks include: forced browsing, command insertion, cross site scripting, buffer overflows, format string attacks, SQL injection, cookie poisoning, and hidden field manipulation.
Attackers can use these flaws to attack backend components through a web application.
The best way to prevent parameter tampering is to ensure that all parameters are validated before they are used. A centralized component or library is likely to be the most effective, as the code performing the checking should all be in one place. Each parameter should be checked against a strict format that specifies exactly what input will be allowed. "Negative" approaches that involve filtering out certain bad input or approaches that rely on signatures are not likely to be effective and may be difficult to maintain.

Total number of alerts in this category: 6

Alerts in this category

Cross Site Scripting

This script is possibly vulnerable to Cross Site Scripting (XSS) attacks.

Cross site scripting (also referred to as XSS) is a vulnerability that allows an attacker to send malicious code (usually in the form of Javascript) to another user. Because a browser cannot know if the script should be trusted or not, it will execute the script in the user context allowing the attacker to access any cookies or session tokens retained by the browser.

Affected item	**/test/actions/do.php**
Affected parameter	param
Variants	2
Affected item	**/test/actions/phpinfo.php**
Affected parameter	
Variants	1

Cross Site Scripting in URI

This script is possibly vulnerable to Cross Site Scripting (XSS) attacks.

Cross site scripting (also referred to as XSS) is a vulnerability that allows an attacker to send malicious code (usually in the form of Javascript) to another user. Because a browser cannot know if the script should be trusted or not, it will execute the script in the user context allowing the attacker to access any cookies or session tokens retained by the browser.

This XSS variant usually appears when a PHP script is using one of following variables without filtering them:

- PHP_SELF
- REQUEST_URI
- SCRIPT_URL
- SCRIPT_URI

Those variables are set either by Apache or the PHP engine. Apache is automatically ignoring anything in the URI after the .php extension for mapping script filename, but these variables are containing the full URI.

Affected item	**/test/actions/phpinfo.php**
Affected parameter	

Variants	2

Cookie manipulation

This script is vulnerable to Cookie manipulation attacks.

By injecting a custom HTTP header or by injecting a META tag is possible to alter the cookies stored in the browser. Attackers will normally manipulate cookie values to fraudulently authenticate themselves on a web site.

Affected item	**/test/actions/do.php**
Affected parameter	param
Variants	1

(Requirement 6.5.2) Broken Access Control

Access control, sometimes called authorization, is how a web application grants access to content and functions to some users and not others. These checks are performed after authentication, and govern what authorized users are allowed to do. Access control sounds like a simple problem but is insidiously difficult to implement correctly. A web application's access control model is closely tied to the content and functions that the site provides. In addition, the users may fall into a number of groups or roles with different abilities or privileges.
Many of these flawed access control schemes are not difficult to discover and exploit. Frequently, all that is required is to craft a request for functions or content that should not be granted. Once a flaw is discovered, the consequences of a flawed access control scheme can be devastating. In addition to viewing unauthorized content, an attacker might be able to change or delete content, perform unauthorized functions, or even take over site administration.

Acunetix WVS did not found any alerts in this category.

(Requirement 6.5.3) Broken Authentication and Session Management

Authentication and session management includes all aspects of handling user authentication and managing active sessions. Authentication is a critical aspect of this process, but even solid authentication mechanisms can be undermined by flawed credential management functions, including password change, forgot my password, remember my password, account update, and other related functions.
User authentication on the web typically involves the use of a userid and password. Stronger methods of authentication are commercially available such as software and hardware based cryptographic tokens or biometrics, but such mechanisms are cost prohibitive for most web applications. A wide array of account and session management flaws can result in the compromise of user or system administration accounts.
Unless all authentication credentials and session identifiers are protected with SSL at all times and protected against disclosure from other flaws, such as cross site scripting, an attacker can hijack a user's session and assume their identity.

Total number of alerts in this category: 5

Alerts in this category

Cross Site Scripting

This script is possibly vulnerable to Cross Site Scripting (XSS) attacks.

Cross site scripting (also referred to as XSS) is a vulnerability that allows an attacker to send malicious code (usually in the form of Javascript) to another user. Because a browser cannot know if the script should be trusted or not, it will execute the script in the user context allowing the attacker to access any cookies or session tokens retained by the browser.

Affected item	**/test/actions/do.php**
Affected parameter	param
Variants	2
Affected item	**/test/actions/phpinfo.php**
Affected parameter	
Variants	1

Cross Site Scripting in URI

This script is possibly vulnerable to Cross Site Scripting (XSS) attacks.

Cross site scripting (also referred to as XSS) is a vulnerability that allows an attacker to send malicious code (usually in the form of Javascript) to another user. Because a browser cannot know if the script should be trusted or not, it will execute the script in the user context allowing the attacker to access any cookies or session tokens retained by the browser.

This XSS variant usually appears when a PHP script is using one of following variables without filtering them:

- PHP_SELF
- REQUEST_URI
- SCRIPT_URL
- SCRIPT_URI

Those variables are set either by Apache or the PHP engine. Apache is automatically ignoring anything in the URI after the .php extension for mapping script filename, but these variables are containing the full URI.

Affected item	**/test/actions/phpinfo.php**
Affected parameter	
Variants	2

(Requirement 6.5.4) Cross Site Scripting (XSS) Flaws

Cross-site scripting (sometimes referred to as XSS) vulnerabilities occur when an attacker uses a web application to send malicious code, generally in the form of a script, to a different end user. These flaws are quite widespread and occur anywhere a web application uses input from a user in the output it generates without validating it.
An attacker can use cross site scripting to send malicious script to an unsuspecting user. The end user's browser has no way to know that the script should not be trusted, and will execute the script. Because it thinks the script came from a trusted source, the malicious script can access any cookies, session tokens, or other sensitive information retained by your browser and used with that site. These scripts can even rewrite the content of the HTML page.
XSS attacks can generally be categorized into two categories: stored and reflected. Stored attacks are those where the injected code is permanently stored on the target servers, such as in a database, in a message forum, visitor log, comment field, etc. The victim then retrieves the malicious script from the server when it requests the stored information. Reflected attacks are those where the injected code is reflected off the web server, such as in an error message, search result, or any other response that includes some or all of the input sent to the server as part of the request. Reflected attacks are delivered to victims via another route, such as in an e-mail message, or on some other web server. When a user is tricked into clicking on a malicious link or submitting a specially crafted form, the injected code travels to the vulnerable web server, which reflects the attack back to the user's browser. The browser then executes the code because it came from a "trusted" server.

Total number of alerts in this category: 5

Alerts in this category

Cross Site Scripting

This script is possibly vulnerable to Cross Site Scripting (XSS) attacks.

Cross site scripting (also referred to as XSS) is a vulnerability that allows an attacker to send malicious code (usually in the form of Javascript) to another user. Because a browser cannot know if the script should be trusted or not, it will execute the script in the user context allowing the attacker to access any cookies or session tokens retained by the browser.

Affected item	**/test/actions/do.php**
Affected parameter	param
Variants	2
Affected item	**/test/actions/phpinfo.php**
Affected parameter	
Variants	1

Cross Site Scripting in URI

This script is possibly vulnerable to Cross Site Scripting (XSS) attacks.

Cross site scripting (also referred to as XSS) is a vulnerability that allows an attacker to send malicious code (usually in the form of Javascript) to another user. Because a browser cannot know if the script should be trusted or not, it will execute the script in the user context allowing the attacker to access any cookies or session tokens retained by the browser.

This XSS variant usually appears when a PHP script is using one of following variables without filtering them:

- PHP_SELF
- REQUEST_URI
- SCRIPT_URL
- SCRIPT_URI

Those variables are set either by Apache or the PHP engine. Apache is automatically ignoring anything in the URI after the .php extension for mapping script filename, but these variables are containing the full URI.

Affected item	**/test/actions/phpinfo.php**
Affected parameter	
Variants	2

(Requirement 6.5.5) Buffer Overflows

Attackers use buffer overflows to corrupt the execution stack of a web application. By sending carefully crafted input to a web application, an attacker can cause the web application to execute arbitrary code - effectively taking over the machine. Buffer overflows are not easy to discover and even when one is discovered, it is generally extremely difficult to exploit. Nevertheless, attackers have managed to identify buffer overflows in a staggering array of products and components. Another very similar class of flaws is known as format string attacks.
Buffer overflow flaws can be present in both the web server or application server products that serve the static and dynamic aspects of the site, or the web application itself. Buffer overflows found in widely used server products are likely to become widely known and can pose a significant risk to users of these products. When web applications use libraries, such as a graphics library to generate images, they open themselves to potential buffer overflow attacks.

Acunetix WVS did not found any alerts in this category.

(Requirement 6.5.6) Injection Flaws

Injection flaws allow attackers to relay malicious code through a web application to another system. These attacks include calls to the operating system via system calls, the use of external programs via shell commands, as well as calls to backend databases via SQL (i.e., SQL injection). Whole scripts written in perl, python, and other languages can be injected into poorly designed web applications and executed. Any time a web application uses an interpreter of any type there is a danger of an injection attack.
SQL injection is a particularly widespread and dangerous form of injection. To exploit a SQL injection flaw, the attacker must find a parameter that the web application passes through to a database. By carefully embedding malicious SQL commands into the content of the parameter, the attacker can trick the web application into forwarding a malicious query to the database. These attacks are not difficult to attempt and more tools are emerging that scan for these flaws. The consequences are particularly damaging, as an attacker can obtain, corrupt, or destroy database contents.

Acunetix WVS did not found any alerts in this category.

(Requirement 6.5.7) Improper Error Handling

Improper handling of errors can introduce a variety of security problems for a web site. The most common problem is when detailed internal error messages such as stack traces, database dumps, and error codes are displayed to the user (hacker). These messages reveal implementation details that should never be revealed. Such details can provide hackers important clues on potential flaws in the site and such messages are also disturbing to normal users.
Web applications frequently generate error conditions during normal operation. Out of memory, null pointer exceptions, system call failure, database unavailable, network timeout, and hundreds of other common conditions can cause errors to be generated. These errors must be handled according to a well thought out scheme that will provide a meaningful error message to the user, diagnostic information to the site maintainers, and no useful information to an attacker.

Acunetix WVS did not found any alerts in this category.

(Requirement 6.5.8) Insecure Storage

Most web applications have a need to store sensitive information, either in a database or on a file system somewhere. The information might be passwords, credit card numbers, account records, or proprietary information. Frequently, encryption techniques are used to protect this sensitive information. While encryption has become relatively easy to implement and use, developers still frequently make mistakes while integrating it into a web application. Developers may overestimate the protection gained by using encryption and not be as careful in securing other aspects of the site. A few areas where mistakes are commonly made include:

- Failure to encrypt critical data
- Insecure storage of keys, certificates, and passwords
- Improper storage of secrets in memory
- Poor sources of randomness
- Poor choice of algorithm
- Attempting to invent a new encryption algorithm
- Failure to include support for encryption key changes and other required maintenance procedures

The impact of these weaknesses can be devastating to the security of a website. Encryption is generally used to protect a site's most sensitive assets, which may be totally compromised by a weakness.

Acunetix WVS did not found any alerts in this category.

(Requirement 6.5.9) Denial of Service

Most web servers can handle several hundred concurrent users under normal use. A single attacker can generate enough traffic from a single host to swamp many applications. Load balancing can make these attacks more difficult, but far from impossible, especially if sessions are tied to a particular server. This is a good reason to make an application's session data as small as possible and to make it somewhat difficult to start a new session.
Once an attacker can consume all of some required resource, they can prevent legitimate users from using the system. Some resources that are limited include bandwidth, database connections, disk storage, CPU, memory, threads, or application specific resources. All of these resources can be consumed by attacks that target them. For example, a site that allows unauthenticated users to request message board traffic may start many database queries for each HTTP request they receive. An attacker can easily send so many requests that the database connection pool will get used up and there will be none left to service legitimate users.

Acunetix WVS did not found any alerts in this category.

(Requirement 6.5.10) Insecure Configuration Management

Web server and application server configurations play a key role in the security of a web application. These servers are responsible for serving content and invoking applications that generate content. In addition, many application servers provide a number of services that web applications can use, including data storage, directory services, mail, messaging, and more. Failure to manage the proper configuration of your servers can lead to a wide variety of security problems. Frequently, the web development group is separate from the group operating the site. In fact, there is often a wide gap between those who write the application and those responsible for the operations environment. Web application security concerns often span this gap and require members from both sides of the project to properly ensure the security of a site's application.

Total number of alerts in this category: 12

Alerts in this category

Apache Mod_Rewrite Off-By-One Buffer Overflow Vulnerability

This alert has been generated using only banner information. It may be a false positive.

Apache mod_rewrite is prone to an off-by-one buffer-overflow condition. The vulnerability arising in the mod_rewrite module's ldap scheme handling allows for potential memory corruption when an attacker exploits certain rewrite rules.

Affected Apache versions:

- Apache 1.3.28 - 1.3.36 with mod_rewrite
- Apache 2.2.0 - 2.2.2 with mod_rewrite
- Apache 2.0.46 - 2.0.58 with mod_rewrite

Affected item	**Web Server**
Affected parameter	
Variants	1

PHP HTML Entity Encoder Heap Overflow Vulnerability

This alert has been generated using only banner information. It may be a false positive.

Stefan Esser reported some vulnerabilities in PHP, which can be exploited by malicious people to cause a DoS (Denial of Service) or potentially compromise a vulnerable system.The vulnerabilities are caused due to boundary errors within the "htmlentities()" and "htmlspecialchars()" functions. If a PHP application uses these functions to process user-supplied input, this can be exploited to cause a heap-based buffer overflow by passing specially crafted data to the affected application. Successful exploitation may allow execution of arbitrary code, but requires that the UTF-8 character set is selected. For a detailed explanation of the vulnerability read the referenced article.
Vendor has released PHP 5.2.0 which fixes this issue.

Affected PHP versions (up to 4.4.4/5.1.6).

Affected item	**PHP**
Affected parameter	
Variants	1

PHP version older than 4.4.1

This alert has been generated using only banner information. It may be a false positive.

Multiple vulnerabilities have been reported in PHP, which can be exploited by malicious people to conduct cross-site scripting attacks, bypass certain security restrictions, and potentially compromise a vulnerable system.

Affected PHP versions (up to 4.4.0).

Affected item	**PHP**
Affected parameter	
Variants	1

PHP Zend_Hash_Del_Key_Or_Index vulnerability

This alert has been generated using only banner information. It may be a false positive.

Stefan Esser had discovered a weakness within the depths of the implementation of hashtables in the Zend Engine. This vulnerability affects a large number of PHP applications. It creates large new holes in many popular PHP applications. Additonally many old holes that were disclosed in the past were only fixed by using the unset() statement. Many of these holes are still open if the already existing exploits are changed by adding the correct numerical keys to survive the unset (). For a detailed explanation of the vulnerability read the referenced article.

Affected PHP versions (up to 4.4.2/5.1.3).

Affected item	**PHP**
Affected parameter	
Variants	1

Unfiltered Header Injection in Apache 1.3.34/2.0.57/2.2.1

This version of Apache is vulnerable to HTML injection (including malicious Javascript code) through "Expect" header. Until not it was not classed as security vulnerability as an attacker has no way to influence the Expect header a victim will send to a target site. However, according to Amit Klein's paper: "Forging HTTP request headers with Flash" there is a working cross site scripting (XSS) attack against Apache 1.3.34, 2.0.57 and 2.2.1 (as long as the client browser is IE or Firefox, and it supports Flash 6/7+).

Affected Apache versions (up to 1.3.34/2.0.57/2.2.1).

Affected item	**Web Server**
Affected parameter	
Variants	1

Apache version older than 1.3.34

This alert has been generated using only banner information. It may be a false positive.

Two potential security issues have been fixed in Apache version 1.3.34:

- If a request contains both Transfer-Encoding and Content-Length headers, remove the Content-Length, mitigating some HTTP Request Splitting/Spoofing attacks.
- Added TraceEnable [o|off|extended] per-server directive to alter the behavior of the TRACE method.

Affected Apache versions (up to 1.3.33).

Affected item	**Web Server**
Affected parameter	
Variants	1

PHPinfo page found

This script is using phpinfo() function. This function outputs a large amount of information about the current state of PHP. This includes information about PHP compilation options and extensions, the PHP version, server information and environment (if compiled as a module), the PHP environment, OS version information, paths, master and local values of configuration options, HTTP headers, and the PHP License.

Affected item	**/test/actions/phpinfo.php**
Affected parameter	
Variants	3

Apache version up to 1.3.33 htpasswd local overflow

This alert has been generated using only banner information. It may be a false positive.

A buffer overflow vulnerability exists in the htpasswd utility included with Apache. The vulnerability is due to improper bounds checking when copying user-supplied 'user' data into local buffers.

Affected Apache versions (up to 1.3.33).

Affected item	**Web Server**
Affected parameter	
Variants	1

Possible sensitive files

A possible sensitive file has been found. This file is not directly linked from the website. This check looks for known sensitive files like: password files, configuration files, log files, include files, statistics data, database dumps. Each of those files may help an attacker to learn more about his target.

Affected item	**/test/actions/test.html**
Affected parameter	
Variants	1

TRACE Method Enabled

HTTP TRACE method is enabled on this web server. In the presence of other cross-domain vulnerabilities in web browsers, sensitive header information could be read from any domains that support the HTTP TRACE method.

Affected item	**Web Server**
Affected parameter	
Variants	1

Affected Items: A Detailed Report

This section provides full details of the types of vulnerabilities found according to individual affected items.

/test/actions/do.php

Cross Site Scripting

This script is possibly vulnerable to Cross Site Scripting (XSS) attacks.

Cross site scripting (also referred to as XSS) is a vulnerability that allows an attacker to send malicious code (usually in the form of Javascript) to another user. Because a browser cannot know if the script should be trusted or not, it will execute the script in the user context allowing the attacker to access any cookies or session tokens retained by the browser.

This alert belongs to the following categories: Requirement 6.5.1, Requirement 6.5.3, Requirement 6.5.4

Parameter	Variations
param	2

Cookie manipulation

This script is vulnerable to Cookie manipulation attacks.

By injecting a custom HTTP header or by injecting a META tag is possible to alter the cookies stored in the browser. Attackers will normally manipulate cookie values to fraudulently authenticate themselves on a web site.

This alert belongs to the following categories: Requirement 6.5.1

Parameter	Variations
param	1

/test/actions/phpinfo.php

Cross Site Scripting

This script is possibly vulnerable to Cross Site Scripting (XSS) attacks.

Cross site scripting (also referred to as XSS) is a vulnerability that allows an attacker to send malicious code (usually in the form of Javascript) to another user. Because a browser cannot know if the script should be trusted or not, it will execute the script in the user context allowing the attacker to access any cookies or session tokens retained by the browser.

This alert belongs to the following categories: Requirement 6.5.1, Requirement 6.5.3, Requirement 6.5.4

Parameter	Variations
	1

Cross Site Scripting in URI

This script is possibly vulnerable to Cross Site Scripting (XSS) attacks.

Cross site scripting (also referred to as XSS) is a vulnerability that allows an attacker to send malicious code (usually in the form of Javascript) to another user. Because a browser cannot know if the script should be trusted or not, it will execute the script in the user context allowing the attacker to access any cookies or session tokens retained by the browser.

This XSS variant usually appears when a PHP script is using one of following variables without filtering them:

- PHP_SELF
- REQUEST_URI
- SCRIPT_URL
- SCRIPT_URI

Those variables are set either by Apache or the PHP engine. Apache is automatically ignoring anything in the URI after the .php extension for mapping script filename, but these variables are containing the full URI.

This alert belongs to the following categories: Requirement 6.5.1, Requirement 6.5.3, Requirement 6.5.4

Parameter	Variations
	2

PHPinfo page found

This script is using phpinfo() function. This function outputs a large amount of information about the current state of PHP. This includes information about PHP compilation options and extensions, the PHP version, server information and environment (if compiled as a module), the PHP environment, OS version information, paths, master and local values of configuration options, HTTP headers, and the PHP License.

This alert belongs to the following categories: Requirement 6.5.10

Parameter	Variations
	3

/test/actions/test.html

Possible sensitive files

A possible sensitive file has been found. This file is not directly linked from the website. This check looks for known sensitive files like: password files, configuration files, log files, include files, statistics data, database dumps. Each of those files may help an attacker to learn more about his target.

This alert belongs to the following categories: Requirement 2.2.4, Requirement 6.5.10

Parameter	Variations
	1

PHP

PHP HTML Entity Encoder Heap Overflow Vulnerability

This alert has been generated using only banner information. It may be a false positive.

Stefan Esser reported some vulnerabilities in PHP, which can be exploited by malicious people to cause a DoS (Denial of Service) or potentially compromise a vulnerable system.The vulnerabilities are caused due to boundary errors within the "htmlentities()" and "htmlspecialchars()" functions. If a PHP application uses these functions to process user-supplied input, this can be exploited to cause a heap-based buffer overflow by passing specially crafted data to the affected application. Successful exploitation may allow execution of arbitrary code, but requires that the UTF-8 character set is selected. For a detailed explanation of the vulnerability read the referenced article.
Vendor has released PHP 5.2.0 which fixes this issue.

Affected PHP versions (up to 4.4.4/5.1.6).

This alert belongs to the following categories: Requirement 6, Requirement 6.5.10

Parameter	Variations
	1

PHP version older than 4.4.1

This alert has been generated using only banner information. It may be a false positive.

Multiple vulnerabilities have been reported in PHP, which can be exploited by malicious people to conduct cross-site scripting attacks, bypass certain security restrictions, and potentially compromise a vulnerable system.

Affected PHP versions (up to 4.4.0).

This alert belongs to the following categories: Requirement 6, Requirement 6.5.10

Parameter	Variations
	1

PHP Zend_Hash_Del_Key_Or_Index vulnerability

This alert has been generated using only banner information. It may be a false positive.

Stefan Esser had discovered a weakness within the depths of the implementation of hashtables in the Zend Engine. This vulnerability affects a large number of PHP applications. It creates large new holes in many popular PHP applications. Additonally many old holes that were disclosed in the past were only fixed by using the unset() statement. Many of these holes are still open if the already existing exploits are changed by adding the correct numerical keys to survive the unset (). For a detailed explanation of the vulnerability read the referenced article.

Affected PHP versions (up to 4.4.2/5.1.3).

This alert belongs to the following categories: Requirement 6, Requirement 6.5.10

Parameter	Variations
	1

Web Server

Apache Mod_Rewrite Off-By-One Buffer Overflow Vulnerability

This alert has been generated using only banner information. It may be a false positive.

Apache mod_rewrite is prone to an off-by-one buffer-overflow condition. The vulnerability arising in the mod_rewrite module's ldap scheme handling allows for potential memory corruption when an attacker exploits certain rewrite rules.

Affected Apache versions:

- Apache 1.3.28 - 1.3.36 with mod_rewrite
- Apache 2.2.0 - 2.2.2 with mod_rewrite
- Apache 2.0.46 - 2.0.58 with mod_rewrite

This alert belongs to the following categories: Requirement 6, Requirement 6.5.10

Parameter	Variations
	1

Unfiltered Header Injection in Apache 1.3.34/2.0.57/2.2.1

This version of Apache is vulnerable to HTML injection (including malicious Javascript code) through "Expect" header. Until not it was not classed as security vulnerability as an attacker has no way to influence the Expect header a victim will send to a target site. However, according to Amit Klein's paper: "Forging HTTP request headers with Flash" there is a working cross site scripting (XSS) attack against Apache 1.3.34, 2.0.57 and 2.2.1 (as long as the client browser is IE or Firefox, and it supports Flash 6/7+).

Affected Apache versions (up to 1.3.34/2.0.57/2.2.1).

This alert belongs to the following categories: Requirement 6, Requirement 6.5.10

Parameter	Variations
	1

Apache version older than 1.3.34

This alert has been generated using only banner information. It may be a false positive.

Two potential security issues have been fixed in Apache version 1.3.34:

- If a request contains both Transfer-Encoding and Content-Length headers, remove the Content-Length, mitigating some HTTP Request Splitting/Spoofing attacks.
- Added TraceEnable [o|off|extended] per-server directive to alter the behavior of the TRACE method.

Affected Apache versions (up to 1.3.33).

This alert belongs to the following categories: Requirement 6, Requirement 6.5.10

Parameter	Variations
	1

Apache version up to 1.3.33 htpasswd local overflow

This alert has been generated using only banner information. It may be a false positive.

A buffer overflow vulnerability exists in the htpasswd utility included with Apache. The vulnerability is due to improper bounds checking when copying user-supplied 'user' data into local buffers.

Affected Apache versions (up to 1.3.33).

This alert belongs to the following categories: Requirement 6, Requirement 6.5.10

Parameter	Variations
	1

TRACE Method Enabled

HTTP TRACE method is enabled on this web server. In the presence of other cross-domain vulnerabilities in web browsers, sensitive header information could be read from any domains that support the HTTP TRACE method.

This alert belongs to the following categories: Requirement 6, Requirement 6.5.10

Parameter	Variations
	1