phpMyAdmin
Web Application Security Assessment

phpMyAdmin

May 18, 2016 – Version 1.2

Prepared for

Michal Čihař

Gervase Markham

Prepared by

Cara Marie

Valentin Leon

Executive Summary

Synopsis

In the spring of 2016, Mozilla engaged NCC Group to perform a security assessment of phpMyAdmin as part of Mozilla's Secure Open Source (SOS) Fund.[1] phpMyAdmin is a free and open source application that has been one of the defacto tools for managing and maintaining MySQL databases for years. Its wide adoption matched with its potential for misuse, warrants regular review from a security perspective.

This assessment was performed remotely by two NCC Group consultants, Cara Marie and Valentin Leon, over a period of two weeks from April 25th to May 6th. The assessment was conducted as a code-assisted penetration test of phpMyAdmin version 4.6.0 (current release as of this writing). phpMyAdmin provided access to key members of the development team and was committed to making this project a success.

Scope

NCC Group's evaluation included:

- Main application portal: This portal is a MySQL web administration tool. The interface supports user and database management, and a console for direct SQL statement and query execution.
- Setup portal: Provides an interface for phpMyAdmin application configuration, including security, import, export, and SQL query settings.
- Source code: While the focus of the assessment was not purely code review, NCC Group reviewed sections dealing with authentication, input validation, and command execution.

Test environments were created using the precompiled phpMyAdmin packages hosted on https://launchpad.net/~nijel/+archive/ubuntu/phpmyadmin.

Key Findings

The assessment uncovered several application flaws. Some of the more notable were:

- A lack of filtering on user CSV output that could allow an attacker to run arbitrary code on an administrator's computer.
- Improper cookie invalidation that could allow an attacker to unset internal global variables.
- Unauthenticated exposure of the Cross-Site Request Forgery (CSRF) protection token that could allow an attacker to perform various attacks against phpMyAdmin users.
- Several traffic flows that expose sensitive data or make use of plaintext HTTP communications. This could allow an attacker to man-in-the-middle, perform CSRF, or various other attacks against phpMyAdmin users.

Limitations

No major blockers were encountered during this assessment. Testing environments had to be setup by NCC Group consultants but this setup process was simple and automated, which helped the consultant team deliver value on the first day of testing. Source code was available on the GitHub project page and access to internal repositories was granted on the first day of testing.

Security Recommendations

In addition to the recommendations specified in each of the vulnerability details , implementing the following high-level recommendations will allow phpMyAdmin to gain a stronger security stance.

- Update phpMyAdmin to support modern security-related headers – HTTP Strict Transport Security and a stricter Content Security Policy. To enable a stricter Content Security Policies (CSP) (see Vulnerability 002), NCC Group recommends removing all legacy scripts inserted as inline JavaScript and move them to script files separate from the HTML source.
- Restrict all state changing actions to occur via HTTP POST. Allowing state changing requests to be sent via GET can leak sensitive data (see Vulnerability 001) and allow attackers to perform numerous attacks against phpMyAdmin. This violates security best practices.
- Provide users with hardening guides at install time and warn users about potential misconfigurations, to reduce the risk of insecure installations.
- Consider requiring installations to use TLS by default. If users have not already acquired certificates, then the installation process should direct them through the process for acquisition via Let's Encrypt.[2]
- Consider implementing an out-of-date software warning to aid users in maintaining up-to-date and secure phpMyAdmin instances.

[1] https://wiki.mozilla.org/MOSS/Secure_Open_Source
[2] https://letsencrypt.org/

Project Dashboard

Target Metadata

Name	phpMyAdmin
Type	Web application
Version	4.6.0
Platform	PHP

Engagement Data

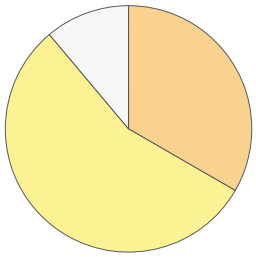
Type	Web application security assessment
Method	Code-assisted
Dates	2016-04-25 to 2016-05-06
Consultants	2
Level of effort	4 person-weeks

Targets

Build site	https://launchpad.net/~nijel/+archive/ubuntu/phpmyadmin
Source	https://github.com/phpmyadmin/phpmyadmin/tree/RELEASE_4_6_0/ – commit 37b38431d167915675fc8ab512470528147e72de

Vulnerability Breakdown

Critical Risk issues	0
High Risk issues	0
Medium Risk issues	3
Low Risk issues	5
Informational issues	1
Total issues	9



Category Breakdown

Authentication	1	
Data Exposure	3	
Data Validation	4	
Session Management	1	

Key

Critical High Medium Low Informational

Table of Vulnerabilities

For each finding, NCC Group uses a composite risk score that takes into account the severity of the risk, application's exposure and user population, technical difficulty of exploitation, and other factors. For an explanation of NCC Group's risk rating and vulnerability categorization, see Appendix A on page 15.

Title	ID	Risk
CSV Export Allows Arbitrary Command Execution in CSV File	006	Medium
Login/Logout Actions Vulnerable to CSRF	007	Medium
Ability to Unset Arbitrary Server Global Variables	009	Medium
Sensitive Values Vulnerable to Session Fixation	005	Low
Sensitive Data in URL GET Query Parameters	001	Low
Overly Permissive Content Security Policy	002	Low
File Traversal Protection Bypass on Error Reporting	004	Low
Self XSS in table_row_action.php	008	Low
Multiple HTTP Plaintext Links	003	Informational

Vulnerability Details

Vulnerability	**CSV Export Allows Arbitrary Command Execution in CSV File**
Risk	**Medium** Impact: Medium, Exploitability: High
Identifier	NCC-1604_MOSS_phpMyAdmin-006
Category	Data Validation
Location	The CSV Export functionality[3]
Impact	A malicious user can change a database field so that when an administrator uses the Export functionality and opens the exported CSV in a spreadsheet editor such as Excel, code may be run on the administrator's computer. Alternatively, a malicious or compromised administrator can add or modify users with formulas in various fields to target other application administrators and users.
Description	The CSV Export functionality does not properly escape exported CSV field values. This can lead to code execution on a user's computer. This is done by including a formula in the CSV file that a spreadsheet editor similar to Excel will evaluate – the formula can include commands to be run on the user's computer. For example, if a user changes a database name or column value to be `=cmd|' /C calc'!A0` and the database is then exported in CSV format by another user, `calc.exe` will run when the file is evaluated by a spreadsheet editor. The victim user will be prompted with a warning, but since the user has downloaded the CSV from a trusted source, they are likely to allow the functionality. It should be noted that encapsulating the values with quotation marks (") does not mitigate the issue.
Reproduction Steps	1. Log into phpMyAdmin and insert `=cmd|' /C calc'!A0` into a database column value. 2. Export the database in CSV format. 3. Open the exported CSV file that was exported in Windows Excel and click through the warnings. 4. Observe the calculator application now running on the computer.
Recommendation	When performing a CSV Export, for any cell that starts with an `=`, `-`, `"`, `@`, or `+`, add a space to the beginning and remove any tab characters (`0x09`) in the cell. Alternatively, prepend each cell field with a single quote, so that their content will be read as text by the spreadsheet editor.

[3] https://github.com/phpmyadmin/phpmyadmin/blob/RELEASE_4_6_0/libraries/plugins/export/ExportCsv.php

Vulnerability	**Login/Logout Actions Vulnerable to CSRF**
Risk	**Medium** Impact: Medium, Exploitability: High
Identifier	NCC-1604_MOSS_phpMyAdmin-007
Category	Data Validation
Location	The phpMyAdmin login[4] and logout[5] functionality
Impact	An attacker could cause a user to logout of phpMyAdmin. Alternatively, a malicious user could cause a phpMyAdmin administrator to authenticate to phpMyAdmin using their account.
Description	Cross-site request forgery (CSRF) countermeasures used in the application are not validated for the login or logout phpMyAdmin functionalities. In a CSRF attack, a user is forced to perform state-changing actions without their knowledge, while authenticated to an application. The current CSRF countermeasures require that all state-changing requests include a `token`. The lack of validation of this value for both the login and logout functionalities leaves users vulnerable to this type of attack. As a result, an attacker could cause a user to authenticate using another account or end their current session.
Reproduction Steps	1. Authenticate to phpMyAdmin. 2. Save the following as *csrf.html*, open it in the same browser. Be sure to modify the `old_usr` value to the authenticated user's name and the domain to reflect your environment if not using a local deployment.

```
<html>
  <body>
    <img src="http://localhost/phpmyadmin/index.php?db=&token=8
        ec5d8124049db01413c49351d0acf44&old_usr=[AUTHENTICATED USER]" />
  </body>
</html>
```

3. Observe the user's ended session.

Recommendation	To ensure that state changing requests are from an authorized user, validate the included `token` value. In addition, to ensure that sensitive data is not leaked via GET URL query parameters perform all state changing requests as POST requests and submitting data via request body (see finding NCC-1604_MOSS_phpMyAdmin-001 on page 10 for further details).

[4] https://github.com/phpmyadmin/phpmyadmin/blob/RELEASE_4_6_0/libraries/common.inc.php#L402-L403
[5] https://github.com/phpmyadmin/phpmyadmin/blob/RELEASE_4_6_0/libraries/common.inc.php#L407-L408

Vulnerability	**Ability to Unset Arbitrary Server Global Variables**
Risk	**Medium** Impact: Undetermined, Exploitability: Medium
Identifier	NCC-1604_MOSS_phpMyAdmin-009
Category	Data Validation
Location	The `PMA_removeRequestVars()` function in cleanup.lib.php[6]
Impact	An attacker can forge requests that will remove (unset) specific global variables on the service side. This can be used for a targeted denial of service (DoS) and potentially other attacks. **Note:** Because of the large amount of global variables, and the relatively short nature of this assessment, NCC Group was unable to fully determine the impact of this vulnerability.
Description	All PHP files include libraries/common.inc.php, which in turn includes libraries/cleanup.lib.php for sanitizing user input if the cross-site request forgery token is missing. The function `PMA_removeRequestVars()` iterates over the user input in GET and POST variables, as well as cookies, and then removes the variables not whitelisted. This function however, appears to be calling `unset()` on `$GLOBAL[$key]` instead of `$_COOKIE[$key]`, effectively removing global variables. An attacker can forge malicious URLs and leverage the cleanup function to remove internal variables. For instance, the `$GLOBAL['cfg']` array, containing the configuration of the application could be targeted. By supplying a parameter named `cfg` in the request, the phpMyAdmin application will be completely inaccessible (local DoS).
Reproduction Steps	Login to phpMyAdmin then access the following URL: /phpmyadmin/index.php?cfg=foo. Notice how the displayed page is completely blank.
Recommendation	Fix the `PMA_removeRequestVars()` function to correctly sanitize the `$_COOKIE` array instead of the `$GLOBAL` array.

[6] https://github.com/phpmyadmin/phpmyadmin/blob/RELEASE_4_6_0/libraries/cleanup.lib.php#L28

Vulnerability	**Sensitive Values Vulnerable to Session Fixation**
Risk	**Low** Impact: Medium, Exploitability: Low
Identifier	NCC-1604_MOSS_phpMyAdmin-005
Category	Session Management
Location	The Cross-Site Request Forgery (CSRF) protection `token` and the `phpMyAdmin` cookie[7]
Impact	An attacker can perform various targeted attacks, such as CSRF, against a victim user or administrator.
Description	Session fixation is a type of session hijacking attack that can occur when a user is not assigned a new session identifier post-authentication. When a user first visits the phpMyAdmin login page, the server sets the `token` CSRF protection value and `phpMyAdmin` cookie. When the user authenticates, these values are not reset. If an attacker knows the value of the user's `token` before the user has logged in, the attacker will be able to perform other targeted attacks, such as cross-site scripting (see finding NCC-1604_MOSS_phpMyAdmin-008 on page 13) or CSRF, against the user. Currently all state-changing and other sensitive actions require additional cookies to successfully execute requests. Consequently, the risk associated with the `phpMyAdmin` cookie session fixation vulnerability is informational – however, if this were to change in future, any action relying on this cookie for validation would be vulnerable to attack.
Reproduction Steps	1. Proxy browser traffic using a proxy similar to Burp Proxy.[8] 2. Visit the phpMyAdmin login page, and make note of the `phpMyAdmin` cookie and CSRF `token` values included in the response. 3. Authenticate to phpMyAdmin. 4. Observe the lack of change for the `phpMyAdmin` cookie and CSRF `token` values.
Recommendation	When users authenticate, change rights levels, or identity, they should immediately be assigned a new session cookie. This can be done by invalidating the old session ID and issuing a new one. In addition, CSRF `tokens` should be reset with each new user session post-authentication.

[7] https://github.com/phpmyadmin/phpmyadmin/blob/RELEASE_4_6_0/libraries/session.inc.php
[8] https://portswigger.net/burp/download.html

Vulnerability **Sensitive Data in URL GET Query Parameters**

Risk **Low** Impact: Medium, Exploitability: Low

Identifier NCC-1604_MOSS_phpMyAdmin-001

Category Data Exposure

Location The following parameters: `token`, `sql_query`, `old_usr`

Impact Sensitive data may be intercepted by an attacker with the ability to read application traffic or logs. The attacker may then be able to perform chosen actions via the user's account, unbeknownst to the user.

Description Several parameters containing sensitive values are transmitted in URL GET query parameters. These parameters include the `token` parameter, used to mitigate cross-site request forgery attacks, `sql_query`, which contains SQL queries that have been run from phpMyAdmin, and `old_usr`, which exposes the previously authenticated account username. It should be noted that while plaintext password values are masked for the `sql_query` parameter, password hashes are not.

Sending sensitive information via URL GET query parameters unnecessarily exposes these values in logs, caches, and `Referer` headers to third parties.

```
GET / HTTP/1.1
Host: codemirror.net
---redacted---
Referer: http://localhost/phpmyadmin/setup/index.php?
token=99b4e72f96146d5904c6f6d00539d711&page=form&formset=Sql_queries
Connection: close
```

`Referer` header `token` leaks are only possible via links from the changelog (/phpmyadmin/changelog.php) or phpMyAdmin setup site (/phpmyadmin/setup/index.php). Neither of these pages make use of url.php,[9] which would effectively prevent this leakage.

While compromise of the `token` parameter can be damaging for a phpMyAdmin user, it is not likely that a third party would abuse the value. It does, however, expand phpMyAdmin's attack surface and puts users at risk in the event of an attack on any of the various third party systems or network traffic.

Reproduction Steps

1. Proxy browser traffic using a proxy similar to Burp Proxy.
2. Authenticate to phpMyAdmin.
3. Observe the `token` as a GET query parameter in the proxy history.

Recommendation Submit the above parameters using POST requests, instead of GET requests. The parameters should be passed in the POST request body and not as URL query parameters.

Update all links from the changelog and phpMyAdmin setup site to use url.php[10] to ensure that data is not unnecessarily exposed to third-party sites.

Do not enable `$cfg['LoginCookieRecall']` by default.[11] This is a defense-in-depth solution to ensure that phpMyAdmin usernames are not leaked via the `old_usr` parameter post logout.

[9] https://github.com/phpmyadmin/phpmyadmin/blob/RELEASE_4_6_0/url.php
[10] https://github.com/phpmyadmin/phpmyadmin/blob/RELEASE_4_6_0/url.php
[11] https://docs.phpmyadmin.net/en/latest/config.html#cfg_LoginCookieRecall

Vulnerability **Overly Permissive Content Security Policy**

Risk **Low** Impact: Low, Exploitability: Low

Identifier NCC-1604_MOSS_phpMyAdmin-002

Category Authentication

Location The Content Security Policy defined in libraries/Header.php[12]

Impact If an application endpoint displays any user input from the URL in the contents of the page, an attacker could craft a link with a malicious JavaScript payload and create a reflected cross-site scripting (XSS) vulnerability. phpMyAdmin is vulnerable, as reflected XSS payloads are being served from page contents inside the origin's domain (which is allowed to run inline JavaScript).

Description Content Security Policy[13] is a security feature that allows web sites to specify settings for modern browsers to enforce protections on web content. In the case of phpMyAdmin, the following `Content-Security-Policy` header allows inline elements such as `<script>` and `<style>` to be included in all pages as long as they are loaded in the context of the same origin.

```
Content-Security-Policy: default-src 'self' ;script-src 'self' 'unsafe-inline'
    'unsafe-eval' ;;style-src 'self' 'unsafe-inline' ;img-src 'self' data:  *.
    tile.openstreetmap.org;
```

Listing 1: Content-Security-Policy Header

While the intention of the policy may have been to restrict loading of script resources to those that originated from the domain, this policy will provide no protection against most types of XSS attacks, as malicious `<script>` injections can still originate from the trusted domain.

Reproduction Steps

1. Proxy browser traffic using a proxy similar to Burp Proxy.
2. Access /phpmyadmin/index.php.
3. Observe the headers returned by the server, in particular, the "Content-Security-Policy" header.

Recommendation Tighten the Content Security Policy as much as possible. Disable `` `unsafe-eval' `` and `` `unsafe-inline' `` if its use cannot be justified.

When adding functionality to the site in the future, keep the scope for allowed scripts, styles, and JavaScript callbacks as small as possible. Remember that Content Security Policy is not a replacement for XSS protection. For more information on the various Content Security Policy directives, see the Mozilla Developer Network.[14]

[12] https://github.com/phpmyadmin/phpmyadmin/blob/RELEASE_4_6_0/libraries/Header.php#L541-L558
[13] https://www.nccgroup.trust/globalassets/our-research/us/whitepapers/csp_best_practices.pdf
[14] https://developer.mozilla.org/en-US/docs/Security/CSP/CSP_policy_directives

Vulnerability File Traversal Protection Bypass on Error Reporting

Risk Low Impact: Low, Exploitability: Low

Identifier NCC-1604_MOSS_phpMyAdmin-004

Category Data Exposure

Location The error reporting endpoint, available at error_report.php[15]

Impact It is possible to confirm the presence of system files on the remote host as well as obtain their line count.

Description The `PMA_countLines()` function available on line 229 of error_report.lib.php[16] contains logic to prevent malicious user input from opening files outside the JavaScript directory. However, it is possible to bypass the protections and access files anywhere on the system.

The function attempts to track the file depth of the path by splitting the path into parts between the slash "/" delimiters. The function decrements the depth on parent "/../" parts, correctly skips dot "/./" parts (no depth change) and increments the depth on everything else. If at anytime the depth becomes negative (meaning that the user reached a directory before the intended root JavaScript directory), the function returns. This logic can by bypassed by sending a path containing empty parts "//", which will increment the tracked depth but remain in the actual folder, allowing an attacker to then use the parent "/../" while still keeping a positive depth.

As the `PMA_countLines()` function only reports the size of the target files and not their contents, the significance of this vulnerability is greatly reduced. Furthermore, to exploit this finding, the *SendErrorReports* configuration setting must be set to *ask* or *always*.

Reproduction Steps

1. Configure phpMyAdmin to report errors by setting the following line in /etc/phpmyadmin/config.inc.php:

```
$cfg['Servers'][$i]['SendErrorReports'] = 'always';
```

2. Proxy browser traffic using a proxy similar to Burp Proxy.
3. Trigger an error in the JavaScript front-end by inserting an incorrect variable or function name in the JavaScript files.
4. Intercept the POST request to /phpmyadmin/error_report.php and modify the following section:

```
exception[stack][12][url]=http://localhost/phpmyadmin/js/get_scripts.
    js.php?scripts[]=////////../../../../../../etc/passwd
```

5. Observe the returned line count.

Recommendation Use PHP's `realpath` function[17] to verify that the target JavaScript directory is in the path.

If this is not possible, process empty parts "//" of the path the same way the dot "/./" path are processed: do not increment the depth. Alternatively, consider rejecting altogether paths containing the parent "/../" directory.

[15] https://github.com/phpmyadmin/phpmyadmin/blob/RELEASE_4_6_0/error_report.php
[16] https://github.com/phpmyadmin/phpmyadmin/blob/RELEASE_4_6_0/libraries/error_report.lib.php#L229-L260
[17] https://secure.php.net/manual/en/function.realpath.php

Vulnerability **Self XSS in table_row_action.php**

Risk **Low** Impact: Low, Exploitability: Low

Identifier NCC-1604_MOSS_phpMyAdmin-008

Category Data Validation

Location The `PMA_getQueryFromSelected()` function in libraries/mult_submits.lib.php[18]

Impact With knowledge of the CSRF token it is possible to trigger a cross-site scripting vulnerability.

Description The `PMA_getQueryFromSelected()` function inside mult_submits.lib.php[19] takes user input and sanitizes it before displaying it back inside the page. However, the function calls `urldecode()` after `htmlspecialchars()`,[20] allowing for special HTML characters to be passed as URL encoded values and displayed back as special characters in the page.

```
$full_query .= 'DELETE FROM '
    . PMA\libraries\Util::backquote(htmlspecialchars($table))
    . ' WHERE ' . urldecode(htmlspecialchars($sval))
    . ';<br />';
```

This vulnerable code is executed when browsing the data of a table, then selecting multiple rows and marking them for deletion. phpMyAdmin will prepare the DELETE SQL statement it is about to execute and display it to the user for confirmation before executing it.

An attacker who is able to gain access to the CSRF token value, such as by exploiting finding NCC-1604_MOSS_phpMyAdmin-005 on page 9, will then be able to exploit this vulnerability and inject JavaScript code into the victim's session.

Reproduction Steps

1. Proxy browser traffic using a proxy similar to Burp Proxy.
2. Navigate to the "Browse" section of a table inside a database.
3. Select some rows then click on "delete".
4. Modify the POST request to /phpmyadmin/tbl_row_action.php with the following body:

```
db=foo&table=bar&token=[token]&goto=sql.php&rows_to_delete%5B0%5D=%2560
bar%2560.%2560a%2560%2B%253D%2B3%25%33%31%25%33%63%25%34%39%25%34%64%25
%34%37%25%32%30%25%35%33%25%35%32%25%34%33%25%33%64%25%32%37%25%36%31%2
5%32%37%25%32%30%25%36%66%25%36%65%25%36%35%25%37%32%25%37%32%25%36%66%
25%37%32%25%33%64%25%32%37%25%36%31%25%36%63%25%36%35%25%37%32%25%37%34
%25%32%38%25%33%31%25%32%39%25%32%37%25%33%65&sql_query=SELECT+*+FROM+%
60bar%60&clause_is_unique=1&ajax_request=true&ajax_page_request=true&su
bmit_mult=delete&_nocache=1462283900030800007a
```

Listing 2: URL encoded payload of "<IMG SRC='a' onerror='alert(1)'>"

5. Notice how the JavaScript payload is executed by the browser.

Recommendation Call `htmlspecialchars()` after calling any type of decoding on user input, in this case, swap the calls to have:

```
    . ' WHERE ' . htmlspecialchars(urldecode($sval))
```

[18] https://github.com/phpmyadmin/phpmyadmin/blob/RELEASE_4_6_0/libraries/mult_submits.lib.php#L477
[19] https://github.com/phpmyadmin/phpmyadmin/blob/RELEASE_4_6_0/libraries/mult_submits.lib.php#L477
[20] https://github.com/phpmyadmin/phpmyadmin/blob/RELEASE_4_6_0/libraries/mult_submits.lib.php#L498

Vulnerability	**Multiple HTTP Plaintext Links**
Risk	**Informational** Impact: Medium, Exploitability: Low
Identifier	NCC-1604_MOSS_phpMyAdmin-003
Category	Data Exposure
Location	Linked pages in the following: • index.php[21] • changelog.php[22] • libraries/config/messages.inc.php[23] • libraries/Util.php[24]
Impact	Requests for the various content linked from phpMyAdmin and the associated setup site may be intercepted by an attacker and modified to include malicious content. Alternatively, an attacker could use this vector to perform an SSL stripping attack against the user.
Description	Multiple pages linked from the phpMyAdmin are hardcoded to make use of plaintext HTTP connections. An attacker in a privileged network position could arbitrarily modify the returned content in order to perform a variety of attacks such as SSL stripping, cross-site scripting, phishing, etc. In addition, pages linked from the changelog (/phpmyadmin/changelog.php) and the phpMyAdmin setup site (/phpmyadmin/setup/index.php) do not make use of url.php.[25] As a result, links from either the changelog or setup site will leak the `token` to third parties via the `Referer` header (see finding NCC-1604_MOSS_phpMyAdmin-001 on page 10).
Reproduction Steps	1. Authenticate to phpMyAdmin. 2. Navigate to the "Official Homepage" link under the "phpMyAdmin" section of homepage. 3. Observe the resulting request via plaintext HTTP.
Recommendation	Update all links to use HTTPS exclusively. It should be noted that each of the cited linked pages support HTTPS versions, which should make securing links relatively painless. Enforcing all links to employ HTTPS will significantly reduce the unnecessary risk placed on users when switching between HTTPS and HTTP connections, and will help mitigate data exposure and man-in-the-middle attacks. Enable HTTP Strict Transport Security (HSTS)[26] to prevent users from accidentally visiting the site over unsecured HTTP and exposing themselves to the risk of an SSL Stripping attack.[27] Note that the HSTS is ignored by browsers when sent over HTTP. It must be sent over HTTPS connections.

[21] https://github.com/phpmyadmin/phpmyadmin/blob/RELEASE_4_6_0/index.php
[22] https://github.com/phpmyadmin/phpmyadmin/blob/RELEASE_4_6_0/changelog.php
[23] https://github.com/phpmyadmin/phpmyadmin/blob/RELEASE_4_6_0/libraries/config/messages.inc.php
[24] https://github.com/phpmyadmin/phpmyadmin/blob/RELEASE_4_6_0/libraries/Util.php
[25] https://github.com/phpmyadmin/phpmyadmin/blob/RELEASE_4_6_0/url.php
[26] https://www.owasp.org/index.php/HTTP_Strict_Transport_Security
[27] https://moxie.org/software/sslstrip/

Appendix A: Vulnerability Field Definitions

The following sections describe the risk rating and category assigned to issues NCC Group identified.

Risk Scale

NCC Group uses a composite risk score that takes into account the severity of the risk, application's exposure and user population, technical difficulty of exploitation, and other factors. The risk rating is NCC Group's recommended prioritization for addressing vulnerabilities. Every organization has a different risk sensitivity, so to some extent these recommendations are more relative than absolute guidelines.

Overall Risk

Overall risk reflects NCC Group's estimation of the risk that a vulnerability poses to the target system or systems. It takes into account the impact of the vulnerability, the difficulty of exploitation, and any other relevant factors.

Critical Implies an immediate, easily accessible threat of total compromise.

High Implies an immediate threat of system compromise, or an easily accessible threat of large-scale breach.

Medium A difficult to exploit threat of large-scale breach, or easy compromise of a small portion of the application.

Low Implies a relatively minor threat to the application.

Informational No immediate threat to the application. May provide suggestions for application improvement, functional issues with the application, or conditions that could later lead to an exploitable vulnerability.

Impact

Impact reflects the effects that successful exploitation upon the target system or systems. It takes into account potential losses of confidentiality, integrity and availability, as well as potential reputational losses.

High Attackers can read or modify all data in a system, execute arbitrary code on the system, or escalate their privileges to superuser level.

Medium Attackers can read or modify some unauthorized data on a system, deny access to that system, or gain significant internal technical information.

Low Attackers can gain small amounts of unauthorized information or slightly degrade system performance. May have a negative public perception of security.

Exploitability

Exploitability reflects the ease with which attackers may exploit a vulnerability. It takes into account the level of access required, availability of exploitation information, requirements relating to social engineering, race conditions, brute forcing, etc, and other impediments to exploitation.

High Attackers can unilaterally exploit the vulnerability without special permissions or significant roadblocks.

Medium Attackers would need to leverage a third party, gain non-public information, exploit a race condition, already have privileged access, or otherwise overcome moderate hurdles in order to exploit the vulnerability.

Low Exploitation requires implausible social engineering, a difficult race condition, guessing difficult to guess data, or is otherwise unlikely.

Category

NCC Group groups vulnerabilities based on the security area to which those vulnerabilities belong. This can help organizations identify gaps in secure development, deployment, patching, etc.

Access Controls	Related to authorization of users, and assessment of rights.
Auditing and Logging	Related to auditing of actions, or logging of problems.
Authentication	Related to the identification of users.
Configuration	Related to security configurations of servers, devices, or software.
Cryptography	Related to mathematical protections for data.
Data Exposure	Related to unintended exposure of sensitive information.
Data Validation	Related to improper reliance on the structure or values of data.
Denial of Service	Related to causing system failure.
Error Reporting	Related to the reporting of error conditions in a secure fashion.
Patching	Related to keeping software up to date.
Session Management	Related to the identification of authenticated users.
Timing	Related to race conditions, locking, or order of operations.