



WP6

DIGIT B1 - EP Pilot Project 645

Deliverable 2: Summary of the Evaluation of Results

Apache Core & APR

Specific contract n°226 under Framework Contract n° DI/07172 – ABCIII

October 2016



Author:



Disclaimer

The information and views set out in this publication are those of the author(s) and do not necessarily reflect the official opinion of the Commission. The content, conclusions and recommendations set out in this publication are elaborated in the specific context of the EU – FOSSA project.

The Commission does not guarantee the accuracy of the data included in this study. All representations, warranties, undertakings and guarantees relating to the report are excluded, particularly concerning – but not limited to – the qualities of the assessed projects and products. Neither the Commission nor any person acting on the Commission's behalf may be held responsible for the use that may be made of the information contained herein.

© European Union, 2016.

Reuse is authorised, without prejudice to the rights of the Commission and of the author(s), provided that the source of the publication is acknowledged. The reuse policy of the European Commission is implemented by a Decision of 12 December 2011.

Report Summary

Title	Summary of the evaluation of results : Apache Core & APR		
Project Owner	Apache Community		
DIGIT Sponsor	EU-FOSSA project		
Author	DIGIT		
Type	Public		
Version	V 0.5	Version date	10/10/2016
Reviewed by	EU-FOSSA Team	Revision date	08/11/2016
Approved by	European Commission - Directorate-General for Informatics (DIGIT)	Approval date	To be approved
		N° Pages	23

Distribution list

Name and surname	Area	Copies
IT contacts	To be identified	To be identified
Communities	Apache security Team	1

Contents

CONTENTS	4
LIST OF TABLES	5
LIST OF FIGURES	6
ACRONYMS AND ABBREVIATIONS	7
1 INTRODUCTION	8
1.1. CONTEXT	8
1.2. SCOPE	9
1.3. DELIVERABLES	9
2 EXECUTIVE SUMMARY	10
3 CODE REVIEW ENVIRONMENT	12
4 SECURITY ASSESMENT	13
4.1. LOW RISK CONTROLS WITH FINDINGS.....	14
4.2. INFORMATIONAL CONTROLS WITH FINDINGS.....	15
4.2.1. <i>Specific C Controls</i>	15
4.2.2. <i>Build Tool (build folder)</i>	16
4.2.3. <i>Findings controlled programmatically</i>	19
5 RECOMMENDATIONS	20
5.1. DETAILS	20
5.2. PRIORITISATION.....	23

List of Tables

Table 1: Security Assessment of CBC-MEM-001	14
Table 2: Security Assessment of CBC-FIO-001.....	15
Table 3: Security Assessment of CBC-VMG-004	15
Table 4: Security Assessment of CBC-VMG-011	16
Table 5: Security Assessment of CBC-MEM-005	17
Table 8: Security Assessment of CBC-SEH-007	18
Table 7: Security Assessment of SCD-FWK-001.....	19
Table 8: Controls with Findings and Recommendations/Specific Solutions	20

List of Figures

Figure 1: Risk Level	10
Figure 2: Priority levels.....	23

Acronyms and Abbreviations

APR	Apache Portable Runtime
CWE	Common Weakness Enumeration
EU-FOSSA	Free and open Source Software Auditing project
FOSS	Free and Open Source Software
IDE	Integrated Development Environment
OS	Operating system
WP	Work Package

1 INTRODUCTION

1.1. Context

The security of the applications used nowadays has become a major concern for organisations, companies and citizens in general, as they are becoming a more common part of our daily lives, and are being used for business and leisure purposes alike. This information has become the most essential asset to protect, as it includes personal information, internal data, industrial property, etc.

From a security point of view, this new scenario presents many new challenges that need to be addressed in order to protect the integrity and confidentiality of the data managed by the applications and their users. Furthermore, their exposure to the Internet has made them a prime target, due to the value that this private and internal information has.

One of the advantages of Free and Open-Source Software (FOSS) is that its source code is readily available for review by anyone, and therefore enables virtually any user to check and provide new features and fixes, including security ones. Also, from a more professional point of view, it allows organisations to review the code completely and find the vulnerabilities or weaknesses that it presents, allowing for a refinement of their security and in turn a safer experience for all the users of the applications.

Objective

The objective of this document is to provide, in a summarised format, the results of the code review ran on the **Apache Core & APR** software. This goes with a set of recommendations focused on increasing the overall security level of the application. This review is carried out within the EU-FOSSA project, focusing on the security aspects of the software.

The objective of this code review is to examine the **Apache Core & APR** software, focusing mainly on its security aspects, the risk that they pose to its users and the integrity and confidentiality of the data contained within.

Apache HTTP Server is one of the most used HTTP and proxy servers and it is FOSS. It is a mature FOSS project running since 1995 and many security flaws have been detected and corrected since its conception.

1.2. Scope

The scope of the project is as follows:

Application name	Apache Core & APR				Review start	25/07/2016
Code review owner	European Commission - Directorate-General for Informatics (DIGIT)				Review end	22/08/2016
Objective	Security Code Review					
Num. Lines	61 286	Version	Apache 2.4.23 APR 1.5.2	Programming language	C	
Code Review Mode	✓	1-Managed	✓	2-Defined	✓	3-Optimised
Libraries	<ul style="list-style-type: none"> Apache Core (version 2.4.23) Apache Portable Runtime (APR, version v1.5.2) 					
Extensions/plugins	N/A					
Services required	N/A					
Result visibility	✓	Internal	✓	Restricted	✓	Public
Critical notification	During assessment			Apache Security Group		
Categories	Data/Input Management	✓	Error Handling / Information Leakage	✓	Specific C controls	✓
	Authentication Controls	✓	Software Communications	✓	Specific C++ controls	✗
	Session Management	✓	Logging/Auditing	✓	Specific JAVA controls	✗
	Authorisation Management	✓	Secure Code Design	✓	Specific PHP controls	✗
	Cryptography	✓	Optimised Mode Controls	✓		

1.3. Deliverables

1 WP6 - Deliverable 1: Code Review Results Report – Apache Core & APR

2 EXECUTIVE SUMMARY

This document presents a high-level report of the code review carried out for the Apache Server (version 2.4.23). As this software includes a large number of optional and third-party modules and extensions, the review has focused on its core: the modules/core module and the Apache Portable Runtime (APR, version v1.5.2) library. The results from this code review, alongside the assessment of any findings identified, will be presented as well. For technical details, please see the complete “Code Review Results Report – Apache Core & APR”¹

This code review has been carried out following a manual review process aided by two open-source review tools:

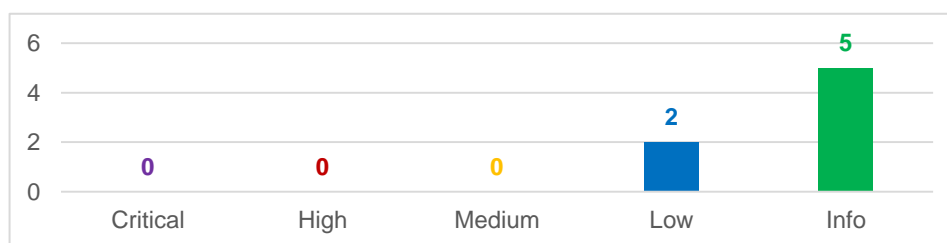
1. **CodeLite**: A Free Open-Source Integrated Development Environment (IDE) for C, it is one of the most used IDE for C and C++, quite easy to install and use.
2. **FlawFinder**: A Free Open-Source code review tool developed by David A. Wheeler, an expert in Free and Open Source Software and secure software development. This tool specialises in finding security flaws in C and C++.

The assessment of the findings pointed out by the code review has been performed from the attackers’ point of view, where:

- The ‘**threat**’ is related to the attacker;
- The ‘**vulnerability**’ is related to the potential issue that may be caused (it means ‘weakness’) and;
- The ‘**impact**’ is related to the consequences of the attack being successful.

Apache Core and APR can be considered mature as far security is concerned, as it is periodically updated/patched and reviewed by the different users. This fact is corroborated if we take a look at the results:

Figure 1: Risk Level



All of the findings can be solved easily without undergoing complex developments, and the risk of them being exploited is either low or not possible without modifying the source code itself.

¹ See the EU-FOSSA Community on Joinup.

Deliverable 2: Summary of the evaluation of results - Apache Code Review

Furthermore, these weaknesses are hard to exploit. This makes it difficult to take advantage of the vulnerabilities in normal environments. However, in custom implementations these needs to be double-checked, as oversights or changes may make these vulnerabilities directly exploitable by attackers.


It is important to notice that this code review does not guarantee that all of the vulnerabilities are detected. Some security issues can remain undetected; therefore it is advisable to carry out other security tests to complement this code review.

As far as the he prioritisation is concerned, it is proposed according to their criticality: low risk findings in the mid-term, and the informative ones in the long-term.


3 CODE REVIEW ENVIRONMENT

In order to carry out the code review and analysis, there was a need for a specific code review environment with the necessary tools (including both automated and manual tools).

For the manual code review, an IDE (Integrated Development Environment) was used:

	<p>CodeLite: a FOSS application that is light, user-friendly and has a high maturity level (version: 9). It is a cross-platform (supporting Windows, the major Linux distributions and Mac OS). It supports the following languages:</p> <ul style="list-style-type: none">• C• C++• JavaScript• PHP <p>One of the main reasons why it was chosen: its excellent support of C and C++ code.</p> <p>Source: http://www.codelite.org/</p>
---	--

Alongside this IDE, an automated tool was also used to help complement the findings and potential results:

	<p>FlawFinder: a FOSS automatic secure code review tool mainly focused on C and C++ code. It supports Linux and Unix-based operating systems mainly, although it can also be run on Windows when compiled using Cygwin. It is compatible with Common Weakness Enumeration (CWE), providing useful feedback on any finding. As a side note, this tool was developed by David A. Wheeler, an authority in the fields of secure software development and open-source software.</p> <p>Source: http://www.dwheeler.com/flawfinder/</p>
---	---

4 SECURITY ASSESSMENT

Findings are identified **in seven controls**. These controls are grouped based on their overall risk level, as follows:

- **Low risk**
 - × *CBC-MEM-001*
 - × *CBC-FIO-001*

- **Informational risk**
 - × *CBC-VMG-004*
 - × *CBC-VMG-011*
 - × *CBC-MEM-005*
 - × *CBC-SEH-007*
 - × *SCD-FWK-001*

4.1. Low Risk Controls with Findings

Table 1: Security Assessment of CBC-MEM-001

CBC-MEM-001	Do not access freed memory		Low
Finding	<p>Legacy Finding: the following finding is mentioned to create awareness among users that keep running Apache servers on older OS (Windows XP, Windows Server 2003...), but it does not have to be fixed as those systems are not supported (by neither Microsoft nor httpd).</p> <p>It does not impact on the Control assessment results.</p>	Threat	Low
		Vulnerability	Medium
		Impact	Low
Detections	File/s:	Line/s:	
	%APR%\threadproc\win32\proc.c	430	
	%APR%\misc\win32\misc.c	223	
	%APR%\locks\win32\thread_cond.c	52	
Assessment	<p>The findings identified within this control are not considered vulnerabilities, as they affect legacy systems not officially supported by Microsoft nor the Apache HTTP project.</p> <ul style="list-style-type: none"> • Threat (Low): it is a publicly known vulnerability. • Vulnerability (Medium): it affects low-memory scenarios in Windows OS. • Impact (Low): it only affects sections of the application related to low-memory scenarios causing instability. <p>Related vulnerability code: N/A.</p>		

Table 2: Security Assessment of CBC-FIO-001

CBC-FIO-001		Exclude user input from format strings		Low
Finding	The printf function is used in the code and it can result in a buffer overflow if the length is not checked.	Threat	Low	
		Vulnerability	Medium	
		Impact	Low	
Detections	File/s:	Line/s:		
	%APR%\misc\win32\misc.c	228		
Assessment	<p>These functions do not provide adequate variable length controls and can result in buffer overflow scenarios.</p> <ul style="list-style-type: none"> • Threat (Low): the string passed to the function is not commonly obtained from direct user input. • Vulnerability (Medium): the lack of length control can be exploited to cause a buffer overflow. • Impact (Low): it would only affect a section of the code, too complex to cause severe damage. <p>Related vulnerability code: CWE-120</p>			

4.2. Informational Controls with Findings

4.2.1. Specific C Controls

Table 3: Security Assessment of CBC-VMG-004

CBC-VMG-004		Do not declare or define a reserved identifier		Info
Finding	The usage of the _MAX suffix in names of variables can lead to conflicts with reserved macros. While it is mostly related to nomenclature formatting, it can lead to the confusion or misuse of the affected variable.	Threat	Low	
		Vulnerability	Low	
		Impact	Low	
Detections	File/s:	Line/s:		
	%APR%\shmem\unix\shm.c	32		
Assessment	<p>The use of the MAX suffix is reserved for macros; using it for other variables or function names can lead to the misuse of said functions if wrongly used in other parts of the code or in extensions/plugins.</p> <ul style="list-style-type: none"> • Threat (Low): the code would need to be modified directly in order to exploit this vulnerability. • Vulnerability (Low): it can compromise the integrity of the data managed by the application. • Impact (Low): it requires direct access to the code and recompilation of the code, would not affect official versions. <p>Related vulnerability code: N/A.</p>			

4.2.2. Build Tool (build folder)

These findings are related to the compilation support libraries that are part of the APR library but take no part on the final executable code generated. The purpose of this library is to assist compilation, therefore the findings are not directly related to the running APR, but to the compilation process. They are included here to serve as a reference for future upgrades and development on them.

Important: these findings do not have a direct impact on the security of the runtime code or on the execution of the server, as they are part of a separate block (build tool) used exclusively during compilation time.

Before deciding to change them, one must take into account the risk of adding more complexity to the code.

Table 4: Security Assessment of CBC-VMG-011

CBC-VMG-011	Do not form or use out-of-bounds pointers or array subscripts		Info
Finding	There is a risk of affecting unexpected memory locations (out of bounds of arrays) or trying to access invalid locations, causing the function involved to crash and cause system instability.	Threat	Low
		Vulnerability	Low
		Impact	Low
Detections	File/s:	Line/s:	
	%APR%\build\jlibtool.c	353	
	%APR%\build\jlibtool.c	341	
	%APR%\tables\apr_hash.c	531	
Assessment	<p>In the code, a decreasing negative loop control variable (loop limit) is used with a function to obtain data from an array.</p> <ul style="list-style-type: none"> • Threat (Low): users cannot directly modify the loop limit, as it is assessed programmatically. • Vulnerability (Low): the risk of losing the integrity of the memory locations managed within the function (or those accessed by it). • Impact (Low): it is complex to exploit this vulnerability, but the lack of a size control of arrays in the code can result in an overflow. <p>Related vulnerability code: N/A.</p>		

Deliverable 2: Summary of the evaluation of results - Apache Code Review

Table 5: Security Assessment of CBC-MEM-005

CBC-MEM-005	Allocate sufficient memory for an object		Info
Finding	The use of the strcpy and strcat functions within the code can lead into a buffer overflow as there is no default control to validate the size of the parameters received.	Threat	Low
		Vulnerability	Low
		Impact	Low
Detections	File/s:	Line/s:	
	%APR%\build\aplibtool.c	157, 272, 850	
Assessment	<p>The strcpy, strcat functions are used within the code reviewed and there are no additional controls to validate the size of the parameters. These calls should be replaced with their updated counterparts (strcpy_s and strcat_s). Several times, memory operations done using memcpy are used without checking the size of the source and destination.</p> <ul style="list-style-type: none"> • Threat (Low): applies only if variable-length strings are used on the section of the code; does not depend on user input. • Vulnerability (Low): can result in a buffer overflow as the size of the string processed (input and output) are not controlled. • Impact (Low): publicly known but complex to execute. <p>Related vulnerability code: CWE-120.</p>		

Table 6: Security Assessment of CBC-SEH-007

CBC-SEH-007	Detect and handle standard library errors		Info
Finding	Several functions are used without checking if an error has taken place, thus they are not managed correctly. These functions are: malloc , remove and fgets .	Threat	Low
		Vulnerability	Low
		Impact	Low
Detections	File/s:	Line/s:	
	%APR%\build\jlibtool.c	325, 969	
	%APR%\build\aplibtool.c	606	
Assessment	<p>A memory allocation (malloc) that can result in a 'NULL' value is not controlled when an error happens. Therefore, any function that depends on this memory allocation will fail during execution if the NULL value is the result.</p> <p>The remove function has to be used with an error checking functionality, so that if an error happens in that line, it should be detected.</p> <p>The fgets function has to be used with an error checking functionality, so that if an error happens in that line, it should be detected.</p> <p>The buffer creation process does not have any measures in order to control the result obtained from the process, which can be a problem if it results in a 'NULL' value due to an error.</p> <ul style="list-style-type: none"> • Threat (Low): it can only be exploited if an attacker is able to trigger errors in those functions (malloc, remove, fgets). • Vulnerability (Low): the results of the use of those functions are not checked against the corresponding error result. • Impact (Low): very complex to exploit, it might modify the software execution lightly. <p>Related vulnerability code: N/A.</p>		

4.2.3. Findings controlled programmatically

Table 7: Security Assessment of SCD-FWK-001

SCD-FWK-001	All frameworks and third party components are up-to-date		Info
Finding	Obsolete functions are used in the code, such as getpass and _alloca .	Threat	Low
		Vulnerability	Medium
		Impact	Low
Detections	File/s:	Line/s:	
	%APR%\password\apr_getpass.c	242	
	%APR%\network_io\win32\sendrecv.c	118	
Assessment	<p>_alloca: In the finding detected in the code, the use of this function in the version evaluated is controlled by ensuring that the parameter is not large enough to cause instability in its use.</p> <p>Taking into account that this function is considered deprecated according to MSDN (for Windows systems) due to the free-up memory controls it provides, it is recommended to consider updating it to use the _malloca function alternative.</p> <p>getpass: In the finding detected in the code, the use of this function in the version evaluated is controlled by ensuring that the function will not be used under Operating Systems in which this function could represent a security flaw.</p> <p>Nevertheless this function is obsolete and not portable. This finding is highlighted in order to keep it in mind for future developments.</p> <p>It is something that adds risk to the code and should be mitigated whenever possible. It is a bad practise to have deprecated or legacy code, as it leads to instability and weaker security, even if it is controlled in its current version. Later versions may override this and raise the finding again. Before deciding to change it, one must take into account the risk of adding more complexity to the code.</p> <ul style="list-style-type: none"> • Threat (Low): it is publicly known and detectable, but can only be exploited indirectly. Nevertheless, it is controlled programmatically. • Vulnerability (Medium): Legacy code is present in the code, nevertheless it is controlled programmatically. • Impact (Low): it only affects a limited part of the application. 		

5 RECOMMENDATIONS

5.1. Details

The code review evaluated the security level of the application analysed and identified vulnerabilities that can put it at risk.

In this section, for each finding a corresponding recommendation is given to help increase the overall security level of the application.

Table 8 shows the recommendations that should be implemented for each of the findings described and assessed in Section 4.

Table 8: Controls with Findings and Recommendations/Specific Solutions

Controls with Findings	Recommendation/Specific Solution
CBC-MEM-001 Do not access freed memory	R01_CBC-MEM-001 These findings only affect implementations of the Apache Server in older operating systems. However, these operating systems are no longer supported by Apache or Microsoft. Furthermore, adding fixes to these legacy findings would introduce complexity to the code and, as it is no longer supported, it is discouraged. Specific Solution: Although it is discouraged to use Apache in older operating systems, and taking into consideration that this should not be fixed by the Apache Foundation, the following information is provided for any older user of legacy OS: Replace <code>InitializeCriticalSection</code> with <code>InitializeCriticalSectionAndSpinCount</code> .
CBC-FIO-001. Exclude user input from format strings	R02_CBC-FIO-001 The use of weak vulnerable functions should be avoided whenever possible as to increase the robustness of the code and prevent related risks as well. Specific Solution: For the case of sprint, it should not be used but replaced with <code>sprintf_s</code> , <code>snprintf</code> , or <code>vsprintf</code> .

Deliverable 2: Summary of the evaluation of results - Apache Code Review

Controls with Findings	Recommendation/Specific Solution
<p>CBC-VMG-004.</p> <p>Do not declare or define a reserved identifier.</p>	<p>R03_CBC-VMG-004</p> <p>Ensure that there is no common variables defined making use of the <code>_MAX</code> suffix, and replace any uses identified. If needed, add controls to ensure that the change does not impact in the code functions that make use of that variable/s.</p>
<p>CBC-VMG-0011.</p> <p>Do not form or use out-of-bounds pointers or array subscripts.</p>	<p>R04_CBC-VMG-011</p> <p>This finding does not have a direct impact on the security of the runtime code, as it is part of a separate block (build tool) used exclusively during compilation time.</p> <p>Before deciding to change it, one must take into account the risk of adding more complexity to the code.</p> <p>Recommendation:</p> <p>Implement control functionality to check the value of the loop limit variable in order to ensure that it is a valid positive number and larger than zero.</p> <p>Any access to arrays (especially within structures) should be done after checking the bounds of that array.</p>
<p>CBC-MEM-005.</p> <p>Allocate sufficient memory for an object</p>	<p>R05_CBC-MEM-005</p> <p>This finding does not have a direct impact on the security of the runtime code, as it is part of a separate block (build tool) used exclusively during compilation time. Before deciding to change it, one must take into account the risk of adding more complexity to the code.</p> <p>Specific Solution: Put in place controls to ensure that the source can be allocated into the destination or:</p> <ul style="list-style-type: none"> ○ Replace all instances of <code>strcpy</code> with <code>strcpy_s</code>. ○ Replace all instances of <code>strcat</code> with <code>strcat_s</code>. <p>Recommendation: The use of <code>memcpy</code> should only be considered after checking the size of the destination memory position against the source, to avoid an overflow.</p>

Deliverable 2: Summary of the evaluation of results - Apache Code Review

Controls with Findings	Recommendation/Specific Solution
<p>CBC-SEH-007.</p> <p>Detect and handle standard library errors</p>	<p>R06_ CBC-SEH-007.</p> <p>This finding does not have a direct impact on the security of the runtime code, as it is part of a separate block (build tool) used exclusively during compilation time. Before deciding to change it, one must take into account the risk of adding more complexity to the code.</p> <p>Recommendation:</p> <ul style="list-style-type: none"> ○ A 'NULL' check should be used after the buffer creation to detect possible errors and handle it properly. ○ A '0' check should be done after using the remove function in order to detect possible errors. ○ A 'NULL' check should be used after using fgets to detect possible errors and handle it properly.
<p>SCD-FWK-001.</p> <p>All frameworks and third party components are up-to-date</p>	<p>R07_ SCD-FWK-001.</p> <p>Despite that this finding is controlled within the code it is included under this section to keep them in mind for future development. Before deciding to change it, one must take into account the risk of adding more complexity to the code.</p> <p>Recommendation: The getpass function is obsolete due to its high insecurity. It should never be used; instead, the functionality should be defined manually in the code to ensure the proper processing of the information according to the needs of the application.</p> <p>Specific Solution: The _alloca function allocates memory on the stack in a Windows system. This function is deprecated because a more secure version is available. The recommendation is to use the new version: _malloca</p>

5.2. Prioritisation

Once the severity of the issues found in the code review has been determined, the following step in the methodology includes a prioritisation process and an action plan definition. This allows the stakeholders and project owners to identify the most urgent findings to solve, allowing the planning of the fixes as part of the standard development cycle.

For this purpose, the following priority sets have been established. The low findings should be tackled in the mid-term, and finally the Informative findings do not require any priority.

Thus, the following graph has been generated:

Figure 2: Priority levels

