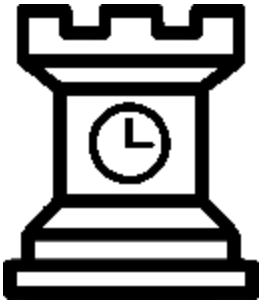
Coverity® Security Report



NTPsec
v. 0.9.1

Enterprise	NTPsec.org
Division	R&D
Assurance Level	AL1
Vignette	Carrier grade
Prepared For	Mark Atwood
Prepared By	Daniel Poirot
Prepared On	Jan 19, 2016, 8:42 PM

Executive Summary

This report details the application security assessment activities that were carried out, providing a summary of findings, compliance against published policy requirements, and remediation actions required. Also provided is a detailed breakdown and cross reference between technical findings and Coverity analysis results.

The intended audience for this report is an application security assurance team and their clients or end users. To review detailed code-level findings, it is recommended that developers click this link to the Coverity Connect platform (https://jasper.local.synopsys.com:8443/reports#p10042) in order to see source code annotated with remediation recommendations.

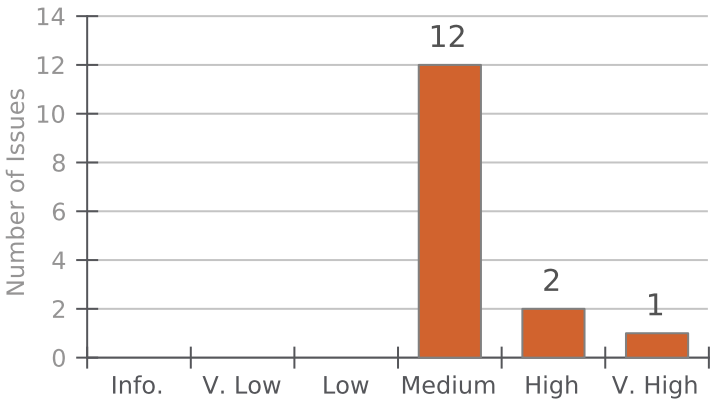
Scorecard

The issues were evaluated according to each element of the report's policy. The results are shown in the table below. An overall status of “pass” is assigned if all the policy elements passed.

Policy Element	Target	Value	Passed
Security Score	90	64	No
OWASP Top 10 Count	0	0	Yes
CWE/SANS Top 25 Count	0	1	No
Analysis Date	20-Dec-15	19-Jan-16	Yes
Overall Status			**No**

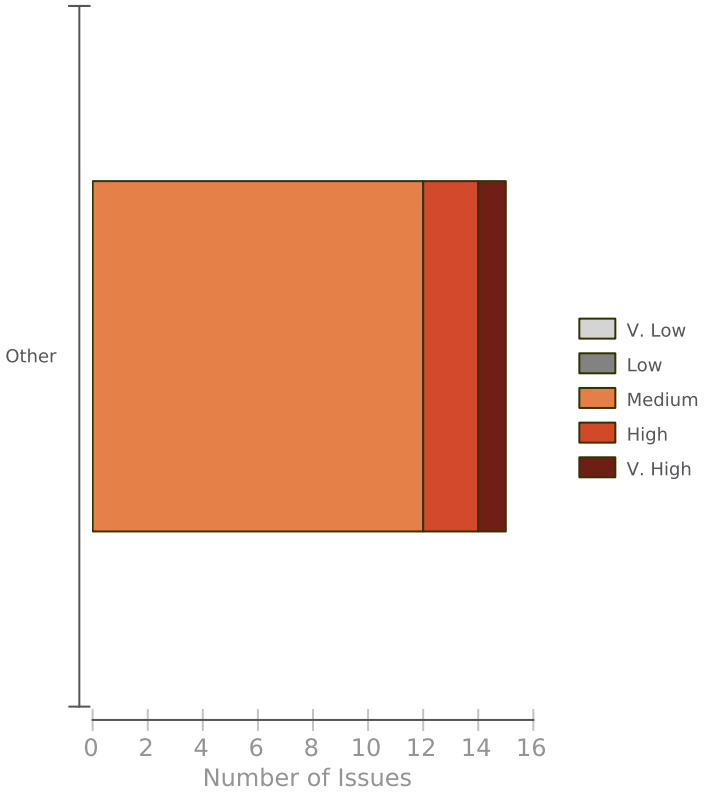
Issues By Severity

A total of 15 security issues were found. Each issue was given a severity based on the vignette. The chart below shows the number of occurrences of each of the six severity values.



Severity By Component

Issues are shown grouped by severity and counted by Component.



Additional Quality Measures

This table reports the numbers of issues of various categories that were not included in the Security Score calculation. Although they were excluded from the report, they may nonetheless indicate the presence of significant quality or security issues.

Category	Count
Issues Marked "False Positive" or "Intentional"	12
Issues Without CWE Numbers	29
Issues Scored as "Informational"	0

Action Items

The code base was evaluated based on the policy in force. The policy has the following elements:
- The Security Score must meet or exceed the target set by the Assurance Level. See the Security Details section for more information.
- There must be no OWASP Top 10 issues among those found in the project. See the OWASP Top 10 section for details.
- There must be no CWE/SANS Top 25 issues among those found in the project. See the CWE/SANS Top 25 section for details.
- All snapshots must have been analyzed within 30 days. See the Analysis Details section for more information.

Coverity recommends the following actions in order to resolve critical outstanding issues, achieve compliance with policy, and improve the overall security of the software.

Security Score Remediation

Resolve issues that contribute to a substandard security score. Resolving the issues below will improve the security score from 64 to 90:
- 2 “High” issues.
- 12 “Medium” issues.

OWASP Top 10 Remediation

The project has no issues in the OWASP Top 10.

CWE/SANS Top 25 Remediation

Resolve 1 issues that are present in the CWE/SANS Top 25. See the CWE/SANS Top 25 Section for a list of them.

Recent Source Code Analysis

Regular source code analysis is key to identifying security issues in a timely manner and to ensuring that these issues are effectively eliminated, in-line with development activities.

The current results are sufficiently recent (less than 30 days old).

Long Term and Residual Risk Management

Review and consider broader improvement to the overall security posture of the target application.
Review outstanding lesser-rated issues to ensure minimal residual risk.
Review issues marked false positive to be sure that a coding change will not eliminate them
Review any security issues marked Informational to see if some are in fact credible threats.
Review and correct non-security issues found by Coverity Analysis, in order to increase the overall quality of the code.

Security Details

The vignette shows how technical impacts (possible security flaws) are paired with severities. This vignette table also shows the number of issues for each technical impact.

Vignette Name: Carrier grade
Vignette Description: Very stringent

Technical Impact	Severity	Number of Issues
Execute unauthorized code	Very High	1
Gain privileges	Very High	0
Modify data	High	0
Denial of service, unreliable execution	High	2
Bypass protection mechanism	High	0
Read data	Medium	0
Denial of service, resource consumption	Medium	12
Hide activities	Medium	0
Total		**15**

Analysis Details

A Coverity project is a collection of one or more streams containing separately-analyzed snapshots. The latest snapshot in each stream is used when reporting results for a project. This section gives details about the streams and the analysis performed for each snapshot.

Stream Name	Snapshot ID	Analysis Date	Analysis Version	Target
NTPsec	10140	19-Jan-16	8.0.0	

The OWASP Top 10 List

The Open Web Application Security Project (OWASP) is an open community dedicated to enabling organizations to conceive, develop, acquire, operate, and maintain applications that can be trusted. The OWASP maintains the OWASP Top 10 List for 2013, a prioritized list of security weaknesses. OWASP says, “We can no longer afford to tolerate relatively simple security problems like those presented in this OWASP Top 10.”

Each entry in the OWASP Top 10 refers to a set of CWE entries. Those entries may be individual weaknesses or families of weaknesses. See the next section for further discussion.

The table below shows the number of issues found in each category of the OWASP Top 10 for 2013.

OWASP Top 10 Category	Count
1. Injection	0
2. Broken Authentication and Session Management	0
3. Cross-Site Scripting	0
4. Insecure Direct Object References	0
5. Security Misconfiguration	0
6. Sensitive Data Exposure	0
7. Missing Function Level Access Control	0
8. Cross-Site Request Forgery	0
9. Using Components with Known Vulnerabilities	0
10. Unvalidated Redirects and Forwards	0
Total	**0**

The CWE/SANS Top 25 List

The Common Weakness Enumeration is a community-developed dictionary of software weakness types. The 2011 CWE/SANS Top 25 Most Dangerous Software Errors (or, “Top 25”) is a list of weaknesses, taken from the CWE, that are thought to be the most widespread and critical errors that can lead to serious vulnerabilities in software.

Each entry in the Top 25 list refers to one CWE identifier. Those identifiers can refer to individual weaknesses or to families of related weaknesses. In contrast, Coverity issues usually refer to the most specific or focused CWE entries. This means that a Coverity issue may be found to be a member of the Top 25 by virtue of this parent/child relationship.

The table below lists all the entries of the Top 25 and shows how many Coverity issues in the current project were found to be members of the Top 25.

CWE/SANS Top 25 Category	CWE Number	Count
1. Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	CWE-89	0
2. Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	CWE-78	0
3. Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')	CWE-120	0
4. Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	CWE-79	0
5. Missing Authentication for Critical Function	CWE-306	0
6. Missing Authorization	CWE-862	0

CWE/SANS Top 25 Category	CWE Number	Count
7. Use of Hard-coded Credentials	CWE-798	0
8. Missing Encryption of Sensitive Data	CWE-311	0
9. Unrestricted Upload of File with Dangerous Type	CWE-434	0
10. Reliance on Untrusted Inputs in a Security Decision	CWE-807	0
11. Execution with Unnecessary Privileges	CWE-250	0
12. Cross-Site Request Forgery (CSRF)	CWE-352	0
13. Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	CWE-22	0
14. Download of Code Without Integrity Check	CWE-494	0
15. Incorrect Authorization	CWE-863	0
16. Inclusion of Functionality from Untrusted Control Sphere	CWE-829	0
17. Incorrect Permission Assignment for Critical Resource	CWE-732	0
18. Use of Potentially Dangerous Function	CWE-676	1
19. Use of a Broken or Risky Cryptographic Algorithm	CWE-327	0
20. Incorrect Calculation of Buffer Size	CWE-131	0
21. Improper Restriction of Excessive Authentication Attempts	CWE-307	0
22. URL Redirection to Untrusted Site ('Open Redirect')	CWE-601	0
23. Uncontrolled Format String	CWE-134	0
24. Integer Overflow or Wraparound	CWE-190	0
25. Use of a One-Way Hash without a Salt	CWE-759	0
Total		**1**

Detailed Issues Ranked By Severity

Severity: Very High

Technical Impact: Execute unauthorized code

CWE 483: Incorrect Block Delimitation

Summary: The code does not explicitly delimit a block that is intended to contain 2 or more statements, creating a logic error.

Details: In some languages, braces (or other delimiters) are optional for blocks. When the delimiter is omitted, it is possible to insert a logic error in which a statement is thought to be in a block but is not. In some cases, the logic error can have security implications.

Remediation: Always use explicit block delimitation and use static-analysis technologies to enforce this practice.

Issue ID (CID) and Issue Type	Source File and Line Number	Component
48812 Nesting level does not match indentation	/libntp/work_thread.c:466	Other

Severity: High

Technical Impact: Denial of service, unreliable execution

CWE 562: Return of Stack Variable Address

Summary: A function returns the address of a stack variable, which will cause unintended program behavior, typically in the form of a crash.

Details: Because local variables are allocated on the stack, when a program returns a pointer to a local variable, it is returning a stack address. A subsequent function call is likely to re-use this same stack address, thereby overwriting the value of the pointer, which no longer corresponds to the same variable since a function's stack frame is invalidated when it returns. At best this will cause the value of the pointer to change unexpectedly. In many cases it causes the program to crash the next time the pointer is dereferenced.

Remediation: Use static analysis tools to spot return of the address of a stack variable.

Issue ID (CID) and Issue Type	Source File and Line Number	Component
48829 Pointer to local outside scope	/libntp/prettydate.c:116	Other

Technical Impact: Denial of service, unreliable execution

CWE 676: Use of Potentially Dangerous Function

This CWE entry is at position 18 in the CWE/SANS Top 25.

Summary: The program invokes a potentially dangerous function that could introduce a vulnerability if it is used incorrectly, but the function can also be used safely.

Remediation: Identify a list of prohibited API functions and prohibit developers from using these functions, providing safer alternatives. In some cases, automatic code analysis tools or the compiler can be instructed to spot use of prohibited functions, such as the "banned.h" include file from Microsoft's SDL. [R.676.1] [R.676.2]

Issue ID (CID) and Issue Type	Source File and Line Number	Component
48784 Calling risky function	/util/sht.c:221	Other

Severity: Medium

Technical Impact: Denial of service, resource consumption

CWE 400: Uncontrolled Resource Consumption ('Resource Exhaustion')

Summary: The software does not properly restrict the size or amount of resources that are requested or influenced by an actor, which can be used to consume more resources than intended.

Details: Limited resources include memory, file system storage, database connection pool entries, or CPU. If an attacker can trigger the allocation of these limited resources, but the number or size of the resources is not controlled, then the attacker could cause a denial of service that consumes all available resources. This would prevent valid users from accessing the software, and it could potentially have an impact on the surrounding environment. For example, a memory exhaustion attack against an application could slow down the application as well as its host operating system.

Remediation: Design throttling mechanisms into the system architecture. The best protection is to limit the amount of resources that an unauthorized user can cause to be expended. A strong authentication and access control model will help prevent such attacks from occurring in the first place. The login application should be protected against DoS attacks as much as possible. Limiting the database access, perhaps by caching result sets, can help minimize the resources expended. To further limit the potential for a DoS attack, consider tracking the rate of requests received from users and blocking requests that exceed a defined rate threshold.

Issue ID (CID) and Issue Type	Source File and Line Number	Component
48834 Large stack use	/util/hist.c:32	Other
48833 Large stack use	/util/hist.c:32	Other
48832 Large stack use	/ntpfrob/jitter.c:75	Other
48831 Large stack use	/ntpfrob/jitter.c:75	Other

Technical Impact: Denial of service, resource consumption

CWE 404: Improper Resource Shutdown or Release

Summary: The program does not release or incorrectly releases a resource before it is made available for re-use.

Details: When a resource is created or allocated, the developer is responsible for properly releasing the resource as well as accounting for all potential paths of expiration or invalidation, such as a set period of time or revocation.

Remediation: Use a language that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.

Issue ID (CID) and Issue Type	Source File and Line Number	Component
48819 Resource leak	/ntpq/ntpq-subs.c:1710	Other
48827 Resource leak	/ntpd/keyword-gen.c:634	Other
48818 Resource leak	/ntpd/ntp_config.c:2441	Other
48820 Resource leak	/libntp/ntp_intres.c:417	Other
48817 Resource leak	/ntpd/ntp_intercept.c:193	Other
48825 Resource leak	/ntpd/keyword-gen.c:552	Other
48816 Resource leak	/ntpd/ntp_intercept.c:300	Other
48826 Resource leak	/ntpd/keyword-gen.c:401	Other

Methodology

Introduction

This report is a distillation of the output of the Coverity Code Advisor used on a particular code source base. Coverity Code Advisor is a static analysis tool that is capable of finding quality defects, security vulnerabilities, and test violations through the process of scanning the output of a specially-compiled code base. The information in this report is specific to security vulnerabilities detected by Coverity Code Advisor and their categorization in the OWASP and CWE/SANS ranking systems.

About Static Analysis

Static analysis is the analysis of software code without executing the compiled program, for the purpose of finding logic errors or security vulnerabilities. Coverity's static analysis tools integrate with all major build systems and generate a high fidelity representation of source code to provide full code path coverage, ensuring that every line of code and execution path is analyzed. Code Advisor supports the market leading compilers for C, C++, Java, C#, Objective C, and Javascript.

About CWE

CWE (Common Weakness Enumeration) is a software community project that is responsible for creating a catalog of software weaknesses and vulnerabilities and is sponsored by the office of Cybersecurity and Communications at the U.S. Department of Homeland Security. The Common Weakness Scoring System (CWSS) provides a method by which to identify and compare weaknesses.

CWE is used by vulnerability-listing efforts such as CWE/SANS Top 25 and OWASP Top 10, among others, to create generalized lists of ranked vulnerabilities. Some, but not all, of the issues reported by Coverity are mapped to CWE-listed vulnerabilities. The Common Weakness Risk Assessment Framework (CWRAF) is a methodology for prioritizing software weaknesses in the context of the software's use. A CWRAF "vignette" prioritizes issues according to their CWE technical impact value. There are 8 technical impacts:

1. modify data,
2. read data,
3. create a denial-of-service that results in unreliable execution,
4. create a denial-of-service that results in resource consumption,
5. execute unauthorized code or commands,
6. gain privileges or assume identity,
7. bypass protection mechanism,
8. hide activities

CWRAF and CWSS allow users to rank classes of weaknesses independent of any particular software package, in order to prioritize them relative to each other.

Setting Priorities with Vignettes

A vignette is a mapping that determines a severity level, or score, for a given technical impact associated with a software issue. This score can in turn be used to derive the priority assigned to remediation of the issue. Coverity provides built-in vignettes to help customers to set these priorities for particular types of applications, and the ability to create custom vignettes.

The part of the vignette that's relevant for this work is the Technical Impact Scorecard. It maps a technical impact to a severity value between Informational (the lowest) and Very High (the highest). This value is known variously as the technical impact's "score" or its "severity". This document uses "severity".

Scoring Methodology

The security score is calculated from the collection of all security-related issues reported by Coverity's code analysis. The score is a number between zero and 100, where 100 is best. The following rules govern the scoring mechanism:

1. If no non-zero severities were found, the score is 100.
2. The most severe event decreases the maximum possible value the score could reach, in proportion to that event's severity.
3. The rest of the events further decrease the score.
4. The effect of the most severe event dominates the score. The effect of the other events is secondary.

The OWASP Top 10 List

The OWASP (Open Web Application Security Project) Foundation is an international organization whose mission is to advance the cause of secure software. As part of its activities, OWASP publishes a report of the most critical web application security flaws in rank order based on the input of a worldwide group of security experts. The most recent version of this list and accompanying report is the OWASP Top 10 List for 2013, The OWSAP Top 10 List is referenced by many standards including MITRE, PCI DSS, DISA, and the FTC.

The CWE/SANS Top 25 List

The SANS Institute is a cooperative research and education organization made up security experts from around the world. SANS is a major source of information on computer security and makes available an extensive collection of research documentation. It also operates the Internet's early security vulnerability warning system, the Internet Storm Center. The 2011 CWE/SANS Top 25 Most Dangerous Software Errors is a list of the most common and critical errors that can lead to software vulnerabilities as published by this organization.

About Coverity

Coverity, a Synopsys company, is a leading provider of quality and security testing solutions. The company, founded in the Computer Science Laboratory at Stanford University, provides an array of tools that assist developers in addressing critical quality and security issues early in the development cycle, thus saving development organizations from remediating issues late in the development cycle or after release when they are much more costly. Many major software development organizations, including 8 of the top 10 global brands and 9 of the top 10 top software companies, deploy Coverity analysis tools. Coverity also maintains a free, cloud based analysis platform, called Scan, for the Open Source Community.