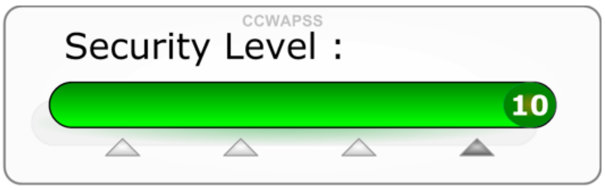
Common Criteria Web Application Security Scoring
CCWAPSS



CCWAPSS 1.1

Author	
Frédéric Charpentier, security pentester. France.	Fcharpentier@xmcopartners.com

Releases	
Version 1.0 : First public release	September 2007
Version 1.1 : Improvements	November 2007

Thanks for theirs contributions and improvement ideas

- Abi Arroyo
- Nabil Ouchn from security-database.com
- Yann Piederriere from cyber-networks.fr
- Ricardo and anonymous people who have left messages on the Ccwapss blog.

1 INTRODUCTION TO CCWAPSS

The purpose of the scoring scale CCWAPSS (to be pronounced [sisiwaps]) is to share a common evaluation method for web application security assessments between security auditors and final customers.

This scale does not aim at replacing other evaluation standards but suggests a simple way to evaluate the security level of a web application.

CCWAPSS is focused on rating the security level of a distinct **web application**, web services or e-business platform. CCWAPSS does not aim at scoring a whole heterogenic perimeter.

1.1 Key benefits of CCWAPSS scoring

Key benefits of CCWAPSS :

- Fighting against the « gaussienne » inclination using a restricted granularity that forces the auditor to **clear-cut** score (there is no medium choice).
- Offering a solution to interpretation problems between different auditors by providing clear and well **documented criteria**.
- The maximum score (10/10) means "compliant with Best Practices". This score could be exceeded in case of excellence (like a medical vision evaluation such as 12/10).
- Each criteria is relative to section of the OWASP Guide 3.0.

1.2 Why develop security level scoring ?

The Common Criteria Web Application Security Scoring has been created by security consultants familiar with web application testing for majors companies.

One of the main questions security auditors have to answer is : "What is our security level on a /10 basis ?".

To answer this question, security auditors needs a scoring methodology which will avoid understanding issues.

The CCWAPSS aims at suggesting such a methodology.

1.3 CCWAPSS delivery

A CCWAPSS score includes :

- The score and its graphical illustration.
- The Evaluation matrix with the 11 criteria and the scoring formula.

A comprehensive *Evaluation matrix* sample are provided in section 3 and *graphics* are provided in section 5.

1.4 CCWAPSS Formula

$$\boxed{\text{Score} = 10 - \sum \text{Risks} + \left(\sum \text{Excellents} / \sum \text{Risks} \right)}$$

With :

Excellent : A positive point is given when the assessed system exceeds the security requirement for a criteria. Refers to section 2.1 "Evaluation".

Risk : Negative points are given when the application does not comply with a criteria. Risk value depends on difficulty of exploit and business impact. Refers to section 2.2 "Risk factor".

The application is evaluated on **11 criteria**. Each criteria could be chosen between 3 possibilities : *Needs Improvement*, *Fair* or *Excellent*.

Criteria evaluated as "Needs improvement" as to be linked with a risk value : 1, 2, 3 or 6.

Evaluation, Risk Factors and criteria are clearly defined in section 2. Frequently Asked Questions are carefully answered in section 4.

1.5 Scoring algorithm

```
Score = 10
E = 0 ; R = 0
For each criteria (out of 10)
    Define Evaluation (Needs Improvement/ Fair / Excellent)
    If "Needs Improvement"
        Define Risk Factor (could this vulnerability lead to major issues?)
        R = R + Risk Factor
    If "Fair"
        Do nothing
    If "Excellent"
        E = E + 1
End of the loop
Score = Score – R + E/R
```

2 SCORING PARAMETERS

2.1 Evaluation

Needs Improvement	Immediate attention should be given to the discussed issues to address significant security exposures. Changes are required.
Fair	Current solution protects the application from security issues. Moderate changes are required to elevate the discussed areas to "Industry Best Practice" standards
Excellent	Exceeds "Industry Best Practice" standards. Security behavior quickly discourages attackers.

2.2 Risk factor

Difficulty of Exploit Vs **Business Impact**	Sophisticated Requires the dedicated effort and time commitment of a skilled attacker or Exploitation of the vulnerability requires a privilege (user account or others) or a knowledge (database name, login name of an user)	Trivial Requires a intermediate skill set and possible use of commonly available tools/knowledge. or Critical vulnerability directly exploitable from a public/anonymous access
Low Limited impact and exposures if the vulnerability is disclosed and exploited.	**1**	**2**
High Significant financial impact, probable negative media exposure, damage to reputation capital.	**3**	**6**

If there are multiple flaws associated to the same criteria, the highest risk factor is kept for the scoring.

2.3 Common criteria

Authentication	
Best Practices	Authentication mechanisms should prevent users without **credentials** from accessing application functionality and prevents against brute-force attacks by restricting smartly block malicious passwords guessing attempts.
Common issues	Login processs bypass, auto-login script, identity switching, …
OWASP Topics:	Section 12 "Guide to Authentication"
Authorization	
Best Practices	Mechanisms should **prevent authenticated users from accessing others users data** or functionalities without the appropriate privileges.
Common issues	Hidden or guessable privileged functionalities, a user A can read/modify data of an user B, GET/POST parameters fuzzing…
OWASP Topics:	Section 13 "Guide to Authorization"
User's Input Sanitization	
Best Practices	**All user-controlled data should be checked** for validity. Bounds checking should be used to prevent buffer overflows and/or variable assignment violations. Syntax checking should be enforced to prevent data encoding, data injection, and/or format string attacks, and to reject forbidden characters. **Output data** generated using users controlled inputs must be sanitized and properly formatted to avoid any client-side script injection through the application.
Common issues	Parameters overflow, JavaScript injection, SQL Injection, Cross Site Scripting, server-side file inclusion, …
OWASP Topics:	Section 21 "Buffer Overflows", Section 15 "Data Validation", Section 16 "Interpreter Injection"
Error Handling and Information leakage	
Best Practices	The application should **trap error messages** that provide detailed system information or application business logic help lead an attacker to compromise the system.
Common issues	SQL verbose errors messages with backend database structures leakage, informative error messages detailing file system path information, logon process notifies whether the account or the password is erroneous, servers banners with release version,…
OWASP Topics:	Section "18 Error Handling", Section 24 "Configuration"

Security Level :
10

Passwords/PIN Complexity	
Best Practices	**Length and complexity requirements** for user authentication should be **enforced** to protect against brute forcing of passwords and PINs.
Common issues	Four digits passwords, common accounts like demo/demo or admin/admin, default installation passwords, …
OWASP Topics:	Section 12 "Guide to Authentication", Section 24 "Configuration"
User's data confidentiality	
Best Practices	The application should maintain **privacy and confidentiality of user's data** throughout the entire data flow lifecycle. Application should ensure confidential data (such as passwords, PINs or MSISDN) are not stored in web server or reverse proxy log files or reside in unencrypted form in cookies or browser cache data. Proper authorization has been implemented to ensure application users are not able to view sensitive information owned by another application user.
Common issues	Credit card numbers and CCV2 code are stored, password appears in clear text in logfiles, administrators can read passwords of others users, logfiles can be read by a unauthorized person, …
OWASP Topics:	Section 8 "Handling E-Commerce Payments", Section 12 "Guide to Authentication"
Session mechanism	
Best Practices	Session management should rely on strong mechanisms and session identifiers. Session identifiers, such as **cookie or session-id**, are difficult to predict, tamper or guess. Session identifiers can not be replayed.
Common issues	Stolen session cookies can be re-used from another IP address. Malicious identity switching by fuzzing parameters.
OWASP Topics:	Section 14 "Session Management", Section 23 "Guide to Cryptography"
Patch management	
Best Practices	All exposed and enabled components of the web application (webapp framework, reverse-proxy, modules, …) must be **up-to-date** regarding the latest **critical security patches** and particularly when exploits are available.
Common issues	A security patch for an exploitable vulnerably is missing and an exploit is in the wild.
OWASP Topics:	Section 27 "Maintenance"

Administration interfaces	
Best Practices	Administration functionality **should be isolated** from the rest of the application. Only authorized users should be allowed to administer the product or application.
Common issues	Common administration URLs, admin functionalities can be called from the main application and profiles, …
OWASP Topics:	Section 22 "Administrative Interface"
Communication security	
Best Practices	Communication of sensitive information should be **encrypted** to prevent unauthorized **eavesdropping** and to ensure data integrity.
Common issues	Passwords are sent encoded with Base64, HTTPS is not used when logon/password or confidential data are posted or received, …
OWASP Topics:	Section 23 "Guide to Cryptography"
Third-Party services exposure	
Best Practices	Any third party dependencies or **others services** that are deployed by default should be heavily audited to ensure that they do not compromise the security of the product or application being supported. Network level filters should be deployed in order to restrict access to third-party services resources or a restrictive rule set must disallow connection from unknown clients.
Common issues	Database listener is accessible from the network, unnecessary services like Finger or RPC are exposed to the clients ...
OWASP Topics:	Section 24 "Configuration", Section 26 "Deployment", Section 19 "File System"

3 EVALUATION MATRIX

The following evaluation matrix is recommended as an attachment to the scoring results. This matrix helps the auditor and the customer to explain the score and to understand evaluation choices.

Criteria	Evaluation **Needs improvement / Fair / Excellent**	Recommendation
Authentication	**Excellent**	
Authorization	**Excellent**	
User's Input Sanitization	**Needs Improvement**	Sanitize input the « id » parameter of the show_all.asp script from caracters : < > % « ` ; // Risk Factor : **3**
Error Handling and Information leakage	**Fair**	
Password/PIN Complexity	**Excellent**	
User's data confidentiality	**Fair**	
Session mechanism	**Fair**	
Communication security	**Needs Improvement**	Use Digest authentication method or use HTTPS when passwords are transmitted. Risk Factor :**1**
Patch management	**Fair**	
Administration interfaces	**Fair**	
Third-Party services exposure	**Needs Improvement**	Restrict access to Oracle TNS service from Internet. Risk Factor : **1**

Score = 10 – (3 + 1 + 1) + ((1+ 1) / 6) = 5.33

If the company modifies the way the webapp handles javascript malicious inputs, the score would be :

Score = 10 - 1 - 1 + ((1 + 1)) / 2 = 8.5

Pentest context :

In this particular case, in addition to the http port, the server exposes an Oracle listener (port 1521). Even if this port is reachable, all exploits and connections are forbidden (that's why the risk factor is lowered to "1"). On the web application, a XSS vulnerability could lead to session robbery if an attacker knows how to send email to users (that 's why the risk factor is on the Sophisticated side). Finally, the login process is a "Basic Authent", which does not protect properly the password. However, this web application is on the Internet (and not on a LAN), so network sniffing probability is limited.

4 FREQUENTLY ASKED QUESTIONS

What is a good score ?

A score of 7 and above is considered to be "good."

Scoring a security level is stupid !

For years, we said that. But when we are delivering pentest results, customers often asks “could you give us a kind score” ? Then, auditor made an evaluation based on instinct and experiences and thus the customers asks “why ?, which criteria ?”.

CCWAPSS is made to respond to the why question and to allow discuss on distinct criteria between clients and auditors.

The score can exceed 10 / 10 !

Yes. If the web application obtains a 10 CCWAPSS score, it simply means that the application is more than compliant with current Best Practices. The scale has been designed in the objective that way : 10 is not a limit. Thus, great secured application can exceed 10.

Regarding the risk factors, why not a “medium” risk ? It’s too binary !

This scale aims at impose the auditor to do clear-cut choices. Otherwise, in a classic low/medium/high scale, the medium choice will often be favored.

How to implement this scale in an automated vulnerability scanner such as Nessus ?

The CCWAPSS scale has been created to be a real smart and to be focused real-world web application risks.

The CCWAPSS scale is not meant to be automatically rated with automated scanners : evaluation have to be performed by a skilled and experienced security auditor.

Even though vulnerability scanners are a great help in the vulnerability assessment process, only human being can made choose between “low or high” or find a vulnerability in the way the application handles privacy between profiles

Difference between Authorization and sessions management ?

Session management criteria is focused on session mechanisms.

If a parameter like “Cookie : id=334T55466; Level=4” can be tampered with “Level=5” and thus the user can access to privileged functions : It’s a session management problem because the problem is due to a weak session mechanism which does not ensure protection against tampering.

In another hand, if a user A can read/modify/call a function/date of a user B without being authorized to, it’s a Authorization problem.

How to choose between Fair and Excellent ?

When the application exceeds the security requirement / Best Practices.

For instance, "Excellent" could be used if the protection is effective and a security warning is prompted to the attacker or if the protection seems "bullet-proof".

How to rate multiple web applications ?

If the web applications are linked (http link, forms redirecting…) : rate together.

If the web applications are not linked by any kind of business/services logic : rate separately.

There is no zero score !

Indeed. Graphics end at 1/10 because we think that delivering a zero score is not constructive regarding the customer and developers. A very low score can often be corrected with simple fixes and tuning. Low scores do not mean that the application must be fully re-engineered.

How to rate when there are multiple flaws associated to the same criteria ?

While auditing a vulnerable web application, the auditor could find more than one flaw which can be rated under the same criteria.

For instance, if an application does not properly enforce authorization controls, the auditor could find lot of different flaws linked to this global lack of control : possibilities to call administrative functions, "jumps" between profiles, authorization bypasses, etc.

There were two ways to handle that kind of situation : **adding** or **majoring**.

The problem with adding flaws is that, when there are more than 2 flaws linked to the same criteria, the CCWAPSS score will quickly go to 1/10 while all these flaws are in fact due to the same problem. And, if the developer fixes this problem, all flaws will be annihilated in a row.

So, CCWAPSS chooses the "majoring" point of view: even though one or more flaws are detected under the same criteria, **the highest will be kept** for the scoring.

Why is auditory criteria missing ?

Logs recording and monitoring are one of the most essential things in IT security. But, CCWAPSS aims to score webapps exclusively regarding the *penetration testing* point of view. So, even if an accurate log system is set, logs will not prevent a hacker from breaking into the webapp. This is why CCWAPS did not used auditory level as an evaluation criteria.

5 GRAPHICS (TO BE CUT AND PASTE)