

# SCARE

**Source Code Analysis Risk Evaluation**

**ISECOM**  
INSTITUTE FOR SECURITY AND OPEN METHODOLOGIES

Any information contained within this document may not be modified or sold without the express consent of ISECOM.  
SCARE for free dissemination under the Open Methodology License 3.0 (OML) and  
CC Creative Commons 2.5 Attribution-NonCommercial-NoDerivs

## Summary

ISECOM, the Institute for Security and Open Methodologies, is registered in Catalonia, Spain as a Non-Profit Organization. Financing for all ISECOM projects has been provided independent of commercial and governmental influence through ISECOM partnerships, subscriptions, certifications, licensing, and case-study-based research.

Your evaluation of this document, suggestions for improvements, and results of its application for further study are required for further development. Contact us at [info@isecom.org](mailto:info@isecom.org) or visit us at [www.isecom.org](http://www.isecom.org) to offer research support, review, and editing assistance.

## Version Information

The current version is SCARE 0.3.

## Restrictions

This research document is free to read, apply and distribute under the Creative Commons 2.5 Attribution-NonCommercial-NoDerivs license.

As a collaborative, open project, SCARE is not to be distributed by any means for which there is commercial gain either by itself or as part of a collection. As a standard, there may be only one, official version of SCARE at any time and that version is not to be altered or forked in any way which will cause confusion as to the purpose of the original methodology. Therefore no derivation of SCARE is allowed.

As a methodology, SCARE is protected under the Open Methodology License 3.0 which applies the protection as that granted to Trade Secrets however where a Trade Secret requires sufficient effort to remain a secret the OML requires the user make sufficient effort to be as transparent as possible about its application. Use and application of the SCARE is acceptance of the responsibility of the user to meet the requirements in the OML. There are no commercial restrictions on the use or application of the SCARE. The OML is available at the end of this manual and at <http://www.isecom.org/oml/>.

Any and all licensing questions or requests can go through ISECOM: [info@isecom.org](mailto:info@isecom.org).

## Contributors

The following people have contributed directly to the development of this methodology and associated tools:

Pete Herzog	<a href="mailto:pete@isecom.org">pete@isecom.org</a>
Marta Barcelo	<a href="mailto:marta@isecom.org">marta@isecom.org</a>
Saulius Grusnys	<a href="mailto:s.grusnys@isagency.eu">s.grusnys@isagency.eu</a>

SCARE has been developed in association with the OpenTC project ([www.opentc.net](http://www.opentc.net)).

## Overview

The Source Code Analysis Risk Evaluation project is a study to create a security complexity metric that will analyze source code and provide a realistic and factual representation of the potential of that source code to create a problematic binary. This metric will not say that the binary will be exploited nor does it do a static analysis for known limitations like vulnerabilities. However it will flag code for a particular interaction type or control and allow the developer to understand which OpSec holes are not protected even if it can't say the effectiveness of that protection. The level of required effectiveness would require a much more sophisticated analysis tool and not within the scope of this project at this time.

The goal of this study is to apply the ISECOM research findings for security metrics represented as the Risk Assessment Values (RAVs) in OSSTMM 3.0. These metrics define "security" as the separation between an asset and a threat. Therefore, Operational Security (OpSec) are the "holes" in the wall of protection, Controls are the patches for those holes, and Limitations are the problems and failures within OpSec and the Controls.

More information regarding the RAVs and OSSTMM 3.0 Security Metrics can be found at <http://www.isecom.org/ravs>.

This computation will provide a final SCARE value, like the RAV, where 100% is the proper balance between controls to OpSec holes and no Limitations. Conversely, less than that shows an imbalance where too few Controls protect OpSec holes or Limitations in OpSec and Controls degrade the security.

Currently, SCARE is designed to work for any programming language. While this methodology shows the C language, we need input and feedback from developers of other languages to expand this further.

# The C Programming Language

## OPERATIONAL SECURITY

This is based on the conclusion of an elemental study that mainly shows there are only two ways to steal something: take it yourself (represented by Access) or have someone else do it for you (represented by Trust). The Visibility is the exposure or knowledge that there is something to steal as for any theft, there is required the opportunity to steal. Therefore OpSec is compromised by these three types. To calculate OpSec, these 3 types are subtracted from the whole.

TYPE	DESCRIPTION	ITEMS
<b>Visibility</b>	The number of files that the program puts or changes on the disk temporarily or permanently collectively during install and run-time.	user data (applications), configuration files (/etc), sensitive user data (e.g. credit card #, software serial #), applications.
<b>Access</b>	The number of places where interactions between a user and the system may occur as part of the input/response interaction between them.	<ol style="list-style-type: none"> <li>1. Direct input from the user from files, system or sockets (gets, fgets, scanf, fscanf, fread, recv, recvfrom, recvmsg).</li> <li>2. Arguments to main() function.</li> <li>3. Environment variables that the user may configure and change.</li> <li>4. Variables which can be passed directly to some other functions, or the program can make copies of those arguments and then manipulate the copies.</li> <li>5. Mathematical actions taken with user provided values.</li> <li>6. Some cases of read() syscall except when the user cannot control the data such as when reading from some device.</li> <li>7. Memory allocation of user-controlled variables</li> <li>8. Directories and file reads where the user may read, create, change, or rename files (readdir, stat, readlink, fstat, lstat).</li> </ol>
<b>Trust</b>	The number of places where interactions between the system, other programs on the system, and the program may occur as part of the input/response interaction between them or within the program itself where that input may be open to manipulation by a user.	<ol style="list-style-type: none"> <li>1. Wherever the user may influence the executed program or the behavior of the other endpoint (exec* , pipe).</li> <li>2. The date and time.</li> <li>3. Where the object is mapped into memory (mmap).</li> </ol>

## CONTROLS

This is based on the 10 controls, of both process and interaction controls, which when all combined can protect to the equivalent of operational security. Limitations in the Controls themselves are counted under Limitations.

TYPE	DESCRIPTION	ITEMS
<b>Authentication</b>	The item interactions are filtered or sanitized according to identified interaction types, role, location, actions, users, data types, or data length.	Items for all these controls still require research and input.
<b>Indemnification</b>	The item interactions provide a warning to the user according to legal statutes.	
<b>Resistance</b>	The item interactions are designed to fail in a manner that does not leave the program or system unsecured in the case of an attack via survivability safeguards.	
<b>Subjugation</b>	The item interactions are controlled by the program in the form of selecting within the range of specific, selectable choices.	
<b>Continuity</b>	The item interactions are designed to continue working regardless of failure via a safeguard, back-up, or redundancy.	
<b>Non-repudiation</b>	Interaction with the item is part of a process which includes recording the identification of the user and the interactions.	

<b>Confidentiality</b>	Interaction with the item is part of a process which includes protecting the data or information between them so as to be understandable only to intended parties, systems or components.	
<b>Privacy</b>	Interaction with the item is part of a process which includes protecting the means of the interaction so that how the interaction takes place is not understandable nor predictable to unintended parties, systems or components.	
<b>Integrity</b>	Interaction with the item is part of a process which includes a means for which if the state or meaning of records is changed that change becomes known.	
<b>Alarm</b>	Interaction with the item is part of a process which includes alerting the program owner when attempts to breach security or circumvent controls are detected.	

## LIMITATIONS

Where OpSec and Controls fail, these are the classifications for their limitations.

TYPE	DESCRIPTION	ITEMS
<b>Vulnerability</b>	The item contains failures related to providing access, denying access, or hiding information/data within the confines of the system.	Items for all these controls still require research and input.

<b>Weakness</b>	The item contains failures related to the Interactive Controls of Authentication, Indemnification, Resistance Subjugation, and Continuity.	
<b>Concern</b>	The item contains failures related to the Process Controls of Non-Repudiation, Confidentiality, Privacy Integrity, and Alarm.	
<b>Exposure</b>	The item contains failures related to Visibility of assets directly or indirectly of interactions.	
<b>Anomaly</b>	The item does not follow programming protocol and will cause the program to act in strange or erratic ways.	