Continuous Mission-Oriented Assessment (CMA) of Assurance[1]

Patrick Hurley, Partha Pal, Mathew Tan Creti, Amy Fedyk
Air Force Research Laboratory, BBN Technologies, Purdue University
patrick.hurley@rl.af.mi, {ppal, afedyk}@bbn.com, mtancret@purdue.edu

Abstract

This paper reports ongoing work on a novel mission-oriented information assurance (IA) assessment approach that contrasts runtime measurements and observations against user-specified requirements.

1. Introduction

It has become routine to have multiple security mechanisms defending advanced mission-critical systems. However, quantifying the contribution of the included mechanisms (e.g., firewalls, antivirus scanners, access control, encryption etc.) or the underlying survivability architecture has so far eluded researchers and practitioners alike. *Information assurance*, i.e., assurance that the security mechanisms are effective, and the system can be entrusted with critical information processing tasks is largely *qualitative*, and is estimated primarily by *offline* analyses, testing, modeling and experimentation. Consequently, during mission execution, arguably the time when it is most critical to be assured about the system, IA takes on an *all-or-nothing* flavor and is mostly dependent on the user's *perception* (i.e., either the user continues to believe the offline assessment or not). Runtime variation of the system's assurance due to attack-induced failures, environmental threats (e.g., release of a new virus), and user-made changes are neither well understood nor considered.

Without a runtime assessment of the system's assurance and given the *perception-based* and *all-or-nothing* view of IA, war fighters and system owners might incorrectly:

- Decide not to use the system fearing compromise,
- Change the system configuration without understanding the impact on the mission, or
- Think the system is protected (and not act on reported events or continue to use) when it is not.

All of these cases incur risks. With the increasing dependence on network centric information systems, these risks cannot be left unaddressed.

In this paper we present initial results of our ongoing work to develop a continuous assessment framework focused on the assurance of mission operations. In this context, a mission refers to a specific set of tasks being executed by an information system to support a group of users cooperating to achieve a common objective, and IA refers to the users' level of confidence that the system can be entrusted with their respective tasks. The high level goal of this research is to demonstrate meaningful and continuous mission-oriented assessment (CMA) of assurance. More specifically, CMA aims to validate the following claim: information systems can be instrumented with suitably placed probes and aggregating mechanisms such that the aggregating mechanisms are able to continuously indicate whether the system is operating at a required level of assurance based on measurements and observations reported by the probes. An additional goal is to support IA assessment-driven adaptive behavior and interoperation with existing QoS mechanisms [1] enabling QoS-IA tradeoffs (e.g., sacrificing encryption for faster response).

The CMA approach is a significant departure from the current thinking of security evaluation. Initial contributions of this early stage research include:

- A taxonomy and organization of factors that contribute to mission-oriented assessment of IA
- A methodology to perform the assessment
- A proof of concept prototype demonstrating mission-oriented continuous assessment and QoS and IA tradeoffs.

2. The CMA Approach

CMA first captures *quantized* levels of IA that are acceptable to the *mission*; then identifies observations and measurements that can be collected from the system and its operating environment; and uses these measurements and observations to determine, on a continuous basis throughout the mission, whether the system is operating within acceptable levels. Apart from measurements that are readily available from existing En-

[1] The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government. Approved for Public Release; Distribution Unlimited: 88ABW-2011-1542, 24Mar11.

terprise System Management (ESM) and security tools, CMA interfaces with other IA and resource management mechanisms to access additional information that is not otherwise visible.

2.1 Multidimensional Assurance State Space

CMA treats acceptable level of assurance as a function of regions in a multi-dimensional state space.

Time is a key dimension in this space because assurance requirements change over time during the mission. Changes in assurance requirements can be based on elapsed time (e.g., for the next 2 hrs since start) or in terms of mission conditions (e.g., from start until an air tasking order is published).

Each mission typically has multiple stakeholders, and each stakeholder typically has his own (time varying) IA requirements. Therefore, *stakeholder* is the second dimension of the multi-dimensional space. We are initially considering three representative classes of stakeholders: *commander* (with an ownership stake),

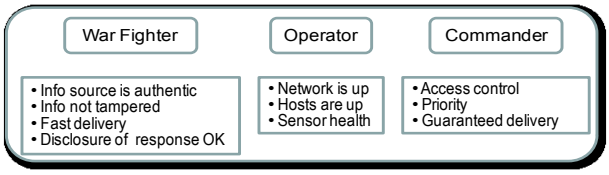

Figure 1: Stakeholders and sample requirements

warfighter (end-user stake) and *operator* (system-administration stakes). Figure 1 shows some examples of stakeholder concerns.

A stakeholder is interested in specific end-to-end capabilities, subsystems and/or subsets of services offered by the system. Therefore, *spatial scope*, capturing the hosts, networks and applications/services that are of interest to a stakeholder is the third dimension.

Classically, security of a system is described in terms of *confidentiality* (C), *integrity* (I), and *availability* (A) [2]. Availability and confidentiality are defined in terms of *authorized* users, but do not consider the strength of authentication and authorization mechanisms involved i.e., whether authorization was based on a user-provided password or validating a common access card (CAC) (CAC authentication being stronger than password-based authentication). CMA includes *strength* of *authentication/access control* (A/A) mechanism as another operationally relevant security attribute independent of and in addition to C, I and A. The *assurance attributes* constitute the remaining dimension of our multi-dimensional assurance space.

We use *projections* to represent and analyze acceptable levels in this high-dimensional assurance space. For example, all stakeholders' interest about a specific assurance attribute (e.g., C) for a specific spatial scope (e.g., an end-to-end interaction between two applications) over time can be captured in a projection

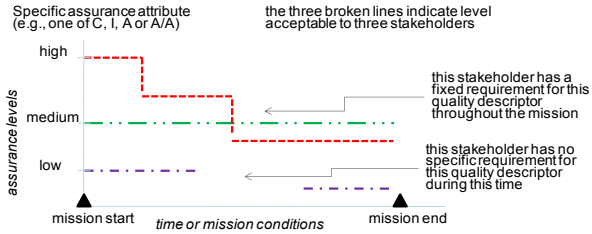

Figure 2: Projection of acceptable levels

like the one shown in Figure 2. To cover all assurance attributes for the given spatial scope four sets of such projections will be needed. In Figure 2, the horizontal axis represents mission progression. Not all projections will have every stakeholder and stakeholders may not have requirements for every point on the time axis.

The required levels of assurance at a given point in the assurance state space are qualitative and ordered. This is deliberate, because while quantifying security is hard even for experts, most stakeholders can qualitatively express their time varying C, I, A, and A/A requirements for the system components and services they use or care about. The following nested loop describes the assurance requirements capture process:

```
For each stakeholder
 For each element in his spatial scope
   For each assurance dimension
     Capture what is acceptable over time
     during the mission in relative terms
```

2.2 Metrics: Measurements and Observations

We argue that no fixed list of metrics is universally applicable and assessment must work with the security mechanisms that are available in the system (i.e., assessment cannot prescribe additional mechanisms). Therefore, the best that can be done is to define the assessment framework in terms of metric classes, and let the assurance engineers choose for each class the actual properties or conditions to measure and observe from what is available in the system at hand. Measurable and observable quantities are generically called "system conditions" (SC). In CMA, the SCs are organized in a shallow hierarchy (Figure 3) with two broad categories *static* and *dynamic*, and leaves (with upper case acronyms) representing the CMA metric classes.

2.2.1 Static System Conditions

The SCs in this category include measurements and observations that can be made even when no mission is currently running. CMA considers two classes of static SCs. The Process and Organizational Maturity (POM) class focuses on the maturity of the software and security engineering process, and the cultural and opera-

tional practices of the organization. The other class in the static category considers Architecture and Infrastructure Quality (AIQ)- how the system is constructed, especially if the system includes mechanisms to protect, detect or manage breaches to basic security properties. In terms of why these classes are relevant for

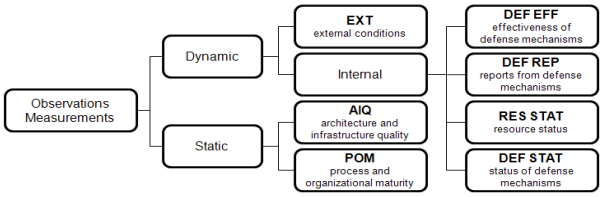

Figure 3: System condition classes

assurance assessment, consider the fact that a large number of current security evaluation methodologies (e.g., NSA INFOSEC Assurance Capability Model (2004), CERT/CC Security Capability Model (2005), NIST SP 800 (2001)) consider process and organizational factors. In previous work [3] we showed that the resiliency against malicious attacks is strongly related to the system's survivability architecture. Some security checklists do include architectural quality and common Weakness Enumeration (CWE) highlights the link between software quality and security assurance.

2.2.2 Dynamic System Conditions

The assurance delivered by the system depends on a number of other factors that can only be measured at runtime. Most of the runtime metrics affecting assurance originate within the system with a few exceptions. Accordingly, the dynamic system conditions are further subdivided into *internal* and *external* SCs.

SCs in the external (EXT) class observe relevant environmental events that impact the system's security. Examples include CERT or vendor-issued advisories, or DEFCON or DHS threat level type indications that may imply increased risk to DoD information systems.

CMA defines 4 *internal* SC classes (right side of Figure-3) to be used in the assessment process. A large segment of such information about the system and its constituent parts (e.g., CPU load, memory load, network load etc.) is already being collected and measured routinely today. However, in typical cases, much of the information gets archived or fed to big board security incident and event management (SIEM) stations where there exists a considerable disconnect between the system operators and stakeholders' requirements.

First, the state of the defense mechanisms (DEF STAT), whether they are functioning at their intended configuration is a primary contributor to the continuous assessment process. The SCs in the DEF STAT class observe and report the configuration state of defense mechanisms in the system. Ideally, changes in defense mechanism configuration resulting from operator action or attack activity should be reflected in the value reported by these SCs. The key challenge to realize this ideal is to make the observing and reporting sufficiently independent of the defense mechanism such that controlling the defense mechanism does not imply controlling the monitor. Enforcement of OS and network-level isolation policies, stronger process authentication and digital signatures can be employed to ensure that the adversary cannot easily feed incorrect observation.

Second, the state of system resources (RES STAT) directly impacts the availability requirements, and since IA mechanisms in general need resources, the ability to meet other security requirements may also be indirectly affected. Therefore, RES STAT SCs, focusing on the status of computing resources, specifically CPU, memory and the network, are important for continuous assessment. Modern hosts and network equipment (e.g., routers) already monitor detailed health statistics. System management tools (e.g., open source Nagios) and protocols (SNMP) that can collect and distribute them efficiently are widely available. SNMP version 3 offers stronger security features such as message integrity, authentication between agent and server, and encryption that the RES STAT SCs can take advantage of.

Third, the DEF REP class includes measurements and observations derived from defense mechanism reports. The DEF REP SCs capture information about unexpected or suspicious incidents and known attack indicators that impact the mission. Survivability architectures or security management tools (e.g., open source OSSEC) already provide a way to interface with host based security mechanisms in a secure way and present processed reports via a server. DEF REP SCs takes advantage of existing mechanisms like OSSEC.

The final class of internal system condition focuses on effectiveness of defense mechanisms (DEF EFF). Modern systems increasingly combine elements of protection, detection and adaptation [4] in their survivability architecture. Assessment of the level of assurance must consider whether defensive responses are being effective. Our prior work [5] on monitoring the effectiveness of defensive responses provides a starting point for the DEF EFF SCs.

2.3 Assurance Assessment Scheme

The first step in employing CMA in a given system involves a thorough system analysis to find out what defense mechanisms are in place, and define the mapping from the assurance levels captured in the projections described in Section 2.1 to what can be observed and measured. This step is analogous to white boarding [6], and will produce a dependency graph (like the one

in Figure 4) showing the spatial scope for each stakeholder and for each assurance attribute of interest along with the list of relevant defense mechanisms. The defense mechanisms can be protection-focused, detection-focused, participate in defensive responses (adaptation-focused) or can be any combination thereof. The subsequent steps are described next.

2.3.1 Identifying What to Observe and Measure

DEF STAT, DEF REP and DEF EFF relate to the defense mechanisms in the graph directly, and RES STAT relates to the spatial scope (i.e., host, services and networks). In CMA, the values presented by the SCs are discrete and ordered, which means raw observations must be processed before reporting.

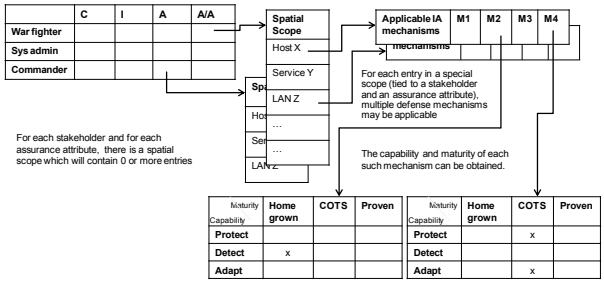

Figure 4: Dependency graph

DEF STAT SCs minimally report the OFF or ON state of defense mechanisms, with the implicit ordering that ON is better than OFF. Each defense mechanism (M1, M2…) present in the dependency graph maps to a DEF STAT SC. If the defense mechanism can operate in *n* configurations (e.g., different key lengths), the corresponding SC projects *n* values, one corresponding to each configuration, with the highest value assigned to the configuration that offers the strongest security (e.g., 128 bit keys are stronger than 64 bit keys).

The value presented by a DEF REP SC reflects the severity of the report produced by the corresponding defense mechanism. Such reports may already include a severity value. Security management tools also process, correlate and rank such reports. Where possible, DEF REP SCs interface with the appropriate mechanisms to avoid custom ranking.

Unlike DEF STAT and DEF REP, DEF EFF SCs are not tied to individual defense mechanisms. In the simplest form, a single DEF EFF SC indicates whether defensive responses mounted by the system are being effective or not. A DEF EFF SC has an internal model of "undesirable" states defined in terms of DEF STAT and RES STAT SCs. When an undesirable state is reached, it starts monitoring DEF REP SCs and operator actions, expecting recovery from the undesired state. In the absence of recovery, the SC begins reporting lower values (i.e., defense is being *ineffective*).

Of the remaining system condition classes, POM is similar to AIQ, and will present a single value. If the system had undergone security evaluation, the POM value should be commensurate with the results. The AIQ SC comes from the maturity and quality of the defense mechanisms and the number of defenses effecting individual assurance attributes (for a given spatial scope for a given stakeholder). There can be multiple SCs in the EXT class, reporting external events such as publication of vulnerability reports. These SCs are identified from additional annotations (not shown in Figure 4) of the assurance dependency graph about the host, OS and application services. For instance, if the system predominantly uses Windows, an EXT system condition may monitor Windows vulnerability reports.

Note that for DEF STAT, DEF EFF, AIQ and POM (i.e., so called *up* SCs) higher value is better; whereas for DEF REP, RES STAT, EXT (i.e., so called *down* SCs) higher is worse. The SCs also do not need to have the same range of values (0-1, 0-5, 1-10 etc.). Some measurements are continuous at the lowest level (e.g. CPU load), but are quantized based on thresholds or reported as a derived discrete measure (e.g., trend of average load crossing a threshold).

In a given system, not all SC classes may be needed or supported. For instance, the deployment environment may not permit live subscription to vulnerability reports (i.e., no EXT SCs). The assurance dependency graph may not be full or exhaustive either. For example, there may not be any defense mechanism covering a specific service whose security is of interest to one or more stakeholders- a system deficiency identified as a byproduct of CMA requirements analysis. CMA is about assessing individual systems, as long as the set of SCs is used in an internally consistent way within the entire assurance space for the mission, such deficiencies are immaterial.

Once the SCs are identified, the system needs to be instrumented so that measurements and observations are projected via system condition objects.

2.3.2 Mapping: Required vs. Supportable Levels

Once the SCs are identified, the next step is to perform a validity check: can the system support the desired number of levels? A stakeholder may expect 3 levels of confidentiality in a link, but the system can be configured to offer only two (say encryption OFF or ON). All projections need to be checked and identified issues need to be reconciled. Because multiple layers of defense are expected in a survivable system, it is likely that the system can be configured to offer more levels than what the stakeholder required. This provides the opportunity to group multiple configurations in a level. As an example, consider a stakeholder's interest in the

confidentiality of an end-to-end capability that uses a link connecting his application with a remote service. The stakeholder's confidentiality requirement is expressed in terms of 3 levels (i.e., High, Medium, Low), and the physical link can either be clear text or encrypted and the application interaction can also be encrypted or clear text (e.g., http or https). Collectively, there are 4 different ways the stakeholder can operate (link unencrypted, http), (link encrypted, http), (link unencrypted, https), (link encrypted, https). In this case we have the option to select which configurations correspond to each of the 3 stakeholder levels. Since the SCs present discrete and ordered values, the above 4 configurations can be described using two binary valued DEF STAT SCs, link_enc and app_enc, as (0,0), (1,0), (0,1) and (1,1) respectively. Because individual system condition values are ordered, these configurations are only partially ordered with (1,1) being the highest and (0,0) being the lowest, with the remaining 2 (i.e., (1,0) and (0,1)) in the middle with no ordering among themselves. One possible mapping of the 3 required levels can be {low = (0,0), medium = (1,0) or (0,1), high = (1,1)}. Alternatively, mission requirements may declare (0,0) unacceptable, which would lead to a grouping like {low=(0,1), medium=(1,0), high= (0,1)}.

2.3.3 General Structure of the Assessment Function

In CMA, the assessment function is the logic that maps the observed SC values to the assurance levels specified by the stakeholders. In general, the assessed level of assurance for stakeholder H, for entity (spatial scope) E of interest to H, and for the security attribute A at any point in the mission is a function of a baseline value and a variance that modifies the baseline:

$$Assessed\ Level_{H,E,A} = L_{H,E,A}(Baseline, Variance)$$

Baseline refers to the (assessment of the) idealized level of assurance that the system is designed to offer in a given configuration. Variance captures the impact of internal changes caused by attacks or user actions as well as external events. Baseline is a function $f(\cdot)$ of DEF STAT SCs, and optionally AIQ, POM, and Variance is a function $g(\cdot)$ of RES STAT system condition, and optionally DEF REP, DEF EFF, EXT, AIQ, POM. In other words:

$$Baseline = f(DEF\ STAT, AIQ, POM)$$
$$Variance = g(RES\ STAT, DEF\ REP, DEF\ EFF, EXT, AIQ, POM)$$

The functions $f(\cdot)$ and $g(\cdot)$ are context dependent to account for mission customizability. DEF STAT and RES STAT are the only mandatory inputs, but other inputs are also accommodated when available. We have already demonstrated meaningful results using only the mandatory inputs.

Evaluating $f(\cdot)$ using DEF STAT amounts to performing a membership operation between the currently observed values of DEF STAT SC (relevant to H, E and A) to the levels (as explained in Section 2.3.2) in the state space of this set of SCs. The variance can be positive or negative – it can either add to or diminish from the baseline assessment depending on the situation. Publication of a new Windows exploit (captured by an EXT system condition) can cause the assessment of all attributes for spatial scopes that include Windows hosts to be lower than the baseline. On the other hand, closing down a vulnerable port on $Host_1$ (captured by a DEF REP system condition) can cause the assessment of all attributes for the spatial scopes that include $Host_1$ to rise. The function $L_{H,E,A}(\cdot,\cdot)$ that combines baseline with variance is also context sensitive. For example, a variance based on RES STAT usually has a multiplicative effect on assessment of availability only, whereas a variance based on EXT e.g., vulnerability report V_R can have a negative additive effect on the assessment of security attributes noted in V_R.

The fact that there is no universal model for computing $f(\cdot)$, $g(\cdot)$ and $L_{H,E,A}(\cdot,\cdot)$ is a continuing challenge. To address this challenge, CMA accommodates user defined context models. As long as the model is consistently used in the system, CMA provides a way to assess the runtime assurance state.

3. Current Prototype

We have an evolving proof of concept assessment framework that we use for demonstrating and evaluating CMA. The aggregator nodes performing assessment for different stakeholders are called the blackboards that communicate with each other using web services. Assessment rules for $f(\cdot)$ etc are defined in TU Prolog, invoked from Java. The first version of the prototype focused on the assessment function and demonstrated continuous assessment for a single stakeholder using only the mandatory DEF STAT and RES STAT SCs in a simulated mission context. Mission progress was simulated by advancing time and by generating mission events. Measurements and observations were changed by injecting values from the simulation control console. The first prototype verified the feasibility of a) the structure and methodology to capture IA requirements, b) the $L_{H,E,A}(\cdot,\cdot)$ assessment scheme, and c) web services-based aggregator interactions. The second version of the prototype deployed blackboards over a distributed set of hosts and demonstrated a) integration with COTS system and security management (e.g., OSSEC and Nagios) to obtain metrics, and b) peering of aggregated information for assessment.

4. Related Work

Many manual IA assessment techniques today rely on vulnerability databases (e.g., CVE [7]) or security policies and guidelines (e.g., EAL levels of common criteria [16]). Vulnerability reports may come from vendors and organizations like CERT. There are repositories like Common Vulnerabilities and Exposures (CVE) [7] that organize and cross-reference reports from disparate sources. Common Vulnerability Scoring System (CVSS) is an example of an extensible standard for vulnerability-based scoring using temporal or environmental factors [8]. On the automated assessment side, Security Content Automation Protocol (SCAP) [9] defines six constituent standards, including CVE and CVSS, for scoring systems based on compliance to specified configuration and the vulnerability impact. In principle, vulnerability-based assessment can be done continuously when the system is in operation and customized for a mission, but it is typically done by an analyst, scoring a specific part or subsystem at a time. In CMA, vulnerability is one of the many factors considered for assessing assurance.

Several NIST publications (e.g., NIST Special Pub. 800-55[10], Federal Information Processing Standard (FIPS) 140 [11]), the orange book and its successor common criteria [12] offer examples of models and rules that are used to assess the security level of information systems or system components. Common Weakness Enumeration (CWE) came out of the need to formally categorize security weaknesses [13], and can also be used in a check-list oriented assessment. But the assessment process needs human experts, is not runtime and does not have a mission focus.

Dependability centric measurements that require long term observation of the system or model-based studies to compute metrics like mean time to attack (MTTA), mean time to failure (MTTF), or mean time to recovery (MTTR) offer another way to rate a system. However, such measures are not suitable for systems that are safety critical, systems that have not been deployed in a comparable environment, and more importantly, for making dynamic and run time adaptation decisions (one of our major goals).

Red teaming [14] is an approach that is often used to assess the quality of defense. While useful to identify system flaws, rating a system based on red team evaluation can be misleading because the capability, motivation and resources of the red team vary.

Quality of protection (QoP) uses *protection* as a quantitative measurement across an entire system. QoP provides an umbrella for a number of different security related properties (e.g., mapping multi-level security to QoP [15], vulnerabilities and best-practices to model QoP [16]). However, the ratings are still static.

5. Conclusion and Next Steps

CMA makes it possible for mission stakeholders to get a continuous status of how the system's IA mechanisms are holding up against their requirements and provides a risk assessment implication of their actions/choices. Even the absence of assessed value (due to disruption in the flow of measurement or observation) is valuable to the stakeholders. CMA also provides the foundation of "managed quality of IA" where IA and service delivery can be actively managed at runtime, trading off IA and QoS as necessary.

Although CMA work is in its early stages, initial results are promising. Informed by early results, we are enhancing the CMA prototype to support additional stakeholders, more complex missions and demonstrate assessments involving additional metric classes. We are also continuously testing and evaluating our prototype. A red team type evaluation to test the CMA approach and meaningful assurance-service delivery tradeoff under adverse conditions is planned.

6. References

[1] Schantz, R., "Quality of Service," *Encyclopedia of Distributed Computing*, Kluwer Academic, 1999.

[2] Avizienis et al. "Fundamental Concepts of Dependability," ISW-2000, Cambridge, MA, 2000.

[3] Pal et al. "The DPASA Survivable JBI- A High-Water Mark in Intrusion-Tolerant Systems", WRAITS workshop, Lisbon, 2007.

[4] Chong et al. "Survivability Architecture of a Mission Critical System: The DPASA Example," ACSAC-21, 2005

[5] Benjamin et al. "Using A Cognitive Architecture to Automate Cyber Defense Reasoning," ECIS BLISS Symposium, Edinburgh, 2008.

[6] Levin, D., "Lessons learned in using live red teams in IA experiments," *DISCEX II*, vol.1, 2003

[7] CVE: http://cve.mitre.org/.

[8] CVSS: http://www.first.org/cvss/.

[9] SCAP: http://scap.nist.gov/.

[10] http://csrc.nist.gov/publications/nistpubs/800-55-Rev1/SP800-55-rev1.pdf

[11] http://csrc.nist.gov/publications/fips/fips1403/fips1403/-Draft.pdf

[12] CC: http://www.commoncriteriaportal.org/thecc.html

[13] CWE: http://cwe.mitre.org/.

[14] Red team: http://idart.sandia.gov/index.html

[15] Foley et al. "Multilevel Security and Quality of Protection," ACM Quality of Protection workshop, 2007.

[16] Aime et.al. "AMBRA: Automated Model-Based Risk Analysis," ACM Quality of Protection workshop, 2007.