MP110439

MITRE PRODUCT

MITRE

The State of Security Automation Standards - 2011

A Survey

Gerard T. McGuire
Emily E. Reid

August, 2011

This page intentionally left blank.

MP110439

MITRE PRODUCT

MITRE

The State of Security Automation Standards - 2011

A Survey

Gerard T. McGuire
Emily E. Reid

August, 2011

Dept. No.: G026

Approved for public release: 11-3822
Distribution unlimited

Abstract

Security automation standards sponsored by the U.S. Government have evolved significantly in the decade since MITRE created and released the Common Vulnerabilities and Exposures (CVE) dictionary. There are now more than two dozen individual standards in use or under development supporting a wide range of security information and functionality. These standards are supported by a variety of sponsors and governance models as well as an ever-growing community of developers, implementers, and users.

Reflective of a growing community, the attendance at NIST's Security Automation Conference has continued to grow over the past several years. The more mature of the standards have been incorporated into hundreds of tools and CVE has become virtually ubiquitous in its subject area. Given the ever-increasing community of adopters, implementers, and contributors, it is clear that the overall security automation effort has been highly successful thus far, and its capabilities and interest in those capabilities continue to grow.

This paper seeks to provide an overview of all the components in security automation as of August 2011.

Acknowledgements

The authors would like to thank the following people for providing information that is encapsulated in this document: Jon Baker, Sean Barnum, Drew Buttner, Brant Cheikes, Steven Christey, Mark Davidson, Tom Graves, Dan Haynes, Bill Heinbockel, Ivan Kirillov, Robert A. Martin, Dave Mann, Joe Sain, Charles Schmidt, Larry Shields, Matthew Wojcik, and John Wunder.

Finally, the authors acknowledge our government sponsors and the many dedicated individuals at National Security Agency (NSA), National Institute of Standards and Technology (NIST), Defense Information Systems Agency (DISA), Department of Homeland Security (DHS), and throughout the security automation community whose efforts have created and maintained this field for the benefit of many. In particular special thanks to Adam Halbardier and David Waltermire for their help in providing information for this report.

Table of Contents

This page intentionally left blank

1 Introduction

This paper provides an overview of the status of security automation standards, use, and community as of August 2011. While it does not cover any particular topic in great depth, it seeks to touch on all significant aspects with sufficient detail to give the reader a general overview of the status of security automation standards. Readers should be able to develop an idea of the coverage of this space, and also find pointers to more detailed information when available.

1.1 Scope

This paper summarizes the status of security automation standards. Specifically, it considers topical standards which have some level of U.S. government involvement or are related to recognized international standards bodies. To be considered, these standards should be under active development, are currently or are intended to be publicly known, and seek to fill a space in the broader sphere of efforts related to standardization of security information and exchange.

This scoping choice excludes several efforts that are interesting and significant, but fail one or more of the above criteria. For example, the Common Malware Enumeration (CME), a CVE-like enumeration of malware, was developed under contract from US-CERT, but is no longer under active development. As such, this effort is not discussed in this document. Additionally, there may be other standards efforts that fit our broad scope but have missed inclusion in this edition of the document.

2 The Security Automation Standards

This section details all security automation standards within the scope of this document. The standards range in maturity from highly mature and stable to conceptual. However, all standards listed here are being actively developed and supported at some level. For each standard we identify the current sponsor, developer, and a URL that serves as the public face of the effort (if any). In addition to the above information, this section provides overviews of each standard.

2.1 The SCAP Standard

Sponsor	Developer	URL
NIST	NIST	http://scap.nist.gov/

The Security Content Automation Protocol (SCAP, pronounced "ess-cap") is a broad program implemented by leveraging a suite of individual standards intended to support the standardization of security measurement and expression to promote interoperability of security products. While the SCAP content leverages seven standards (CVE, CCE, CPE, XCCDF, OVAL, OCIL and CVSS), it is both more and less than the sum of its parts: More because it includes specific guidance for how the individual standards should be used in combined operation; less because it explicitly does not encompass all the features of individual components.

Most of the standards that define the SCAP content predate SCAP itself, which was first introduced (version 1.0) in 2006. There were preceding efforts coordinating between standards for as long as there has been more than one standard, but these efforts were usually limited to single pairs of standards. SCAP provided a valuable service by creating a single repository for housing content based on the standard interoperation of the base standards and is under a central authority.

The first major revision of SCAP was released in February of 2011. SCAP 1.1 primarily extends SCAP 1.0 by including OCIL 2.0 as a requisite standard and updating the OVAL standard to version 5.8 (previously it was version 5.4).

SCAP continues to evolve. Work has commenced on SCAP 1.2, which was released in its first draft form for public comment in July 2011 and closed in August 2011. Some significant additions proposed for SCAP 1.2 include electronic signatures of source and result data streams, the adoption of CPE 2.3, and the addition of the Common Configuration Scoring system (CCSS).

SCAP is primarily concerned with describing interactions and establishing metrics to validate compliance with content written against sets of the component standards. The first version of the validation program requirements (NIST IR 7511) was produced in 2009. The second version of the validation program requirements is (NIST IR 7511 R 2) was published in February 2011.

The rest of this section looks at the individual standards that make up the foundational standards used within SCAP. Each standard is discussed without reference to any limits which SCAP itself might place on its usage in creating SCAP content.

2.1.1 CVE

Sponsor	Developer	URL
DHS	MITRE	http://cve.mitre.org/

The Common Vulnerabilities and Exposures list (CVE) is an enumeration of unique identifiers for publicly known security vulnerabilities. CVE was created under internal research and development funds by MITRE and first published in 1999, making it the oldest of all the standards discussed in this document. The inspiration for this effort came from two internal MITRE tasks: a comparison of vulnerability scanning tools, and an attempt to map information security advisories to specific software vulnerabilities. Both of these efforts were stymied by the inconsistent naming of software vulnerabilities among vendors and researchers. The leaders of the two MITRE tasks realized the need for a common, vendor/researcher-neutral way to refer to individual vulnerabilities. From this initial observation, MITRE created CVE to provide a common name for vulnerabilities to support inter-product comparison and compatibility.

CVE has been a huge success and is actively sponsored by the U.S. Government. There are currently more than 46,000 CVE IDs and over 280 products from over 160 organizations that are either officially CVE Compatible or that have made declarations of compatibility. CVEs are now included in the initial bug reports of almost all major vendors (over 70 organization participating in total) as well as many other vulnerability announcement portals (e.g., ISS, Secunia, US-CERT, etc.) Many would argue that the current breadth of security automation standards owes its existence to the success of the initial CVE program.

2.1.2 OVAL

Sponsor	Developer	URL
DHS	MITRE	http://oval.mitre.org/

The Open Vulnerability and Assessment Language (OVAL) is a language for making logical assertions about the state of an endpoint system. OVAL was developed by MITRE and first released in October of 2002. The motivation for this effort was the observation that, even though vulnerability assessment tools could use CVE IDs to provide a mutually compatible count of vulnerabilities, tools would often disagree as to whether a particular vulnerability was present because they would use different system state artifacts as indicators. OVAL was developed as a way to define a canonical description of the state associated with the presence of a vulnerability. In its original version, OVAL definitions were written in an SQL-like language, but an XML version of the language was quickly added, and in 2004 the SQL version was deprecated.

The OVAL Language consists of several parts:

- Definitions Schema, which is used to identify acceptable system states and characteristics
- System Characteristics Schema, which is used to record the discovered system state
- Results Schema, which records pass/fail findings as determined by the tests in a definition file

Both the OVAL Definitions Schema and OVAL System Characteristics Schema are extended through additional component schemas which define platform-specific structures that are needed

to represent checks of specific system entities or related findings, respectively. For example, the Windows Definition component schema defines the structures used to identify entries in the Windows registry or Active Directory, while the Linux Definition component schema defines structures to identify and evaluate RPM (RPM Package Manager) information. Every type of system artifact that can be located and evaluated by OVAL needs to have appropriate structures defined in both a Definition and System Characteristics component schema.

In addition to the language itself, the OVAL team also maintains an open-source interpreter (the OVAL Definition Interpreter, or OVALDI) for the OVAL Language. This interpreter takes OVAL Definitions as input and evaluates the state of a target system automatically, reporting back the state that it actually found (in the OVAL System characteristics) and whether the discovered state matched the requirements dictated in the OVAL definition (in the OVAL Results). The interpreter is provided as a reference implementation to aid implementers seeking to create their own OVAL tools. The OVALDI does not support all of the component schemas although it does offer extensive support for all structures in the core language as well as the Windows and Linux component schemas. It is primarily intended to serve as an example rather than a full featured OVAL interpreter.

The current version of OVAL is 5.9. OVAL has continued to evolve through several releases. The last four releases have added both low level technical features and a number of high level capabilities. Among the major innovations, OVAL 5.6 added support for a new regular expression language syntax and the ability to evaluate system artifacts against multiple states. OVAL 5.8 added extensive support for Linux, Mac OS, Solaris, UNIX, and Windows platforms, enhanced the OVAL Results directives (allowing for more flexibility in the contents of OVAL Results), and supported more granular OVAL Item collection.

Still pending final release, OVAL 5.10 adds features to support a malware artifact hunting use case and support for Windows PowerShell. The latter provides greater power and flexibility to content authors on Microsoft Windows platforms.

As part of an effort to provide an authoritative source of development information, upcoming versions of OVAL will be released with an accompanying specification.

2.1.2.1 The OVAL Repository

Sponsor	Developer	URL
DHS	MITRE	http://oval.mitre.org/repository/index.html

In addition to the OVAL Language and the OVAL Interpreter, the OVAL team maintains the OVAL Repository. The repository is a public collection of OVAL content including vulnerability, patch and inventory OVAL Definitions. It is a place for the community to develop, share, and discuss a large body of content for use in relevant tools. Repository content is contributed by the community – the MITRE OVAL team generally only performs syntactic review and management of the repository.

Currently the OVAL Repository contains nearly 11,000 definitions and has grown at a rate of approximately 30% annually over the course of the past few years.

2.1.3 XCCDF

Sponsor	Developer	URL
NSA/NIST	MITRE	http://scap.nist.gov/specifications/xccdf/

The eXtensible Configuration Checklist Description Format (XCCDF) was initially developed by NSA and first published in 2004. XCCDF provides a common language to express, organize, and manage security guidance. XCCDF also supports references to checking systems (such as OVAL), allowing an XCCDF interpreter to direct checking tools to perform an automated assessment of a system. Special tailoring capabilities and pre-generated Profiles allow users to customize such an assessment based on broad security postures or individualize based on the needs of their environments. Finally, the XCCDF language can also encapsulate the results of such assessments.

As noted above, the original XCCDF release candidates were published in 2004 and the first full version was published as a NIST Interagency Report (NIST IR 7188) in early 2005. Before September 2011, the most recent revision of XCCDF was version 1.1.4, finalized in early 2008 as NIST IR 7275 r3. Recently, the community has been actively working on changes for the next version, 1.2, which will include several new features such as the ability to report check results individually when multiple checks are executed for a single rule. NIST IR 7275 r4, the XCCDF 1.2 Specification, was released in September 2011.

The XCCDF 1.1.4 Reference Implementation Interpreter, a command-line application demonstrating interpretation of XCCDF Benchmarks, was first released in November 2010. This tool focuses on the automated benchmark use case of XCCDF. A revised release was made public in January 2011, and has resulted in many downloads. In April 2011, a draft release of an interpreter for the new version of XCCDF occurred to demonstrate the functioning of some of the recently accepted features. In October 2011, an XCCDF 1.2 version of the Interpreter was released.

2.1.4 CCE

Sponsor	Developer	URL
NSA	MITRE	http://cce.mitre.org/

The Common Configuration Enumeration (CCE) is an enumeration of security-relevant configuration elements for applications and operating systems. In the initial launch of CVE, which was initially called the Common Vulnerability Enumeration, the need to address misconfiguration-based vulnerabilities led to renaming CVE to its current use of “Exposure” to cover the topics now addressed by CCE. First published in 2006, CCE provides a common name by which configuration elements might be referred, regardless of the ways in which that configuration action might be implemented. For example, on a Windows machine, a particular security feature might be enabled either via a registry change or through a Group Policy Object, but because these both control a single functional feature, they would be associated with a single CCE. CCEs are divided into “platform groups”. A CCE platform group roughly identifies the operating system or application to which a CCE entry applies. CCE’s platform groups adhere to the same level of granularity commonly found in security configuration guidance that are written

for individual platforms, as well as in the sets of checks and other features found in configuration audit and management tools. For example, Microsoft Windows XP and Sun Solaris 10 are CCE platform groups.

In addition to an ID, entries contain a short description, the ways in which the configuration change might be effected (in the previous example, this would identify the relevant registry key and GPO (Group Policy Object)), a list of the settings for the given configuration item (e.g., enable/disable), and a few sample references where the given configuration is mentioned in guidance documents. Entries do not contain any recommendation as to a "correct" setting for any configuration item. This allows CCE entries to be applicable across multiple sets of recommendations, many of which may have differing requirements.

The number of CCE entries currently stands at 10,300, and the CCE team is working with Cisco and Mozilla to generate CCEs for their platforms. The current list of platform groups includes IBM AIX 5.3, HP-UX 11.23, Microsoft IE7 and IE8, Office 2007, Office 2010, RHEL4, RHEL5, Solaris 8, Solaris 9, Solaris 10, Oracle WebLogic Server 11G, Windows 2000, Windows XP, Windows 2003, Windows Vista, Windows Server 2008, Windows Server 2008 R2, and Windows 7. The team is also working on augmenting the CCE supporting infrastructure to increase its ability to respond to requests for additions, updates and corrections.

2.1.5 CPE

Sponsor	Developer	URL
NSA	MITRE	http://cpe.mitre.org/

The Common Platform Enumeration (CPE) is a structured naming scheme used to identify information technology systems, platforms, and packages. CPE is predated by another effort called XCCDF-P, which was active in 2005 and 2006. CPE was inspired by XCCDF-P, but pursued the same goal using a completely different approach. First published in 2007, CPE provides standard product names to ensure that there is a shared understanding as to the software and hardware that is referenced in security recommendations and reports. CPE Names currently take the form of a structured URI with colon-separated components, e.g.,

```
cpe:/a:microsoft:word:2010:sp1:enterprise:en-us
```

As illustrated above, CPE names consist of seven components: *part* (“a” for “application”), *vendor* (“microsoft”), *product name* (“word”), *version* (“2010”), *update* (“sp1” for “service pack 1”), *edition* (“enterprise”), and *language* (“en-us” for US English). The Official CPE dictionary is maintained and managed by NIST,[1] contains over 32,000 CPE names, and receives hundreds of new or modified entries each month.

Version 2.2 of the CPE specification was released in March 2009. Development of a maintenance release—designated version 2.3—began in March 2010 with the formation of the CPE Core team—including MITRE, NIST, DoD, Cisco, McAfee, and nCircle. Drafts of the CPE version 2.3 specifications were released by NIST for public comment in April and June 2011. The specification of CPE v2.3 has been broken out into four separate specification documents: Naming, Matching, Dictionary, and Language (NIST IRs 7695, 7696, 7697, and 7898 respectively). MITRE is the principal author for the Naming and Matching specifications, while NIST authored the Dictionary and Language specifications.

[1] The Official CPE Dictionary can be accessed at http://nvd.nist.gov/cpe.cfm

Version 2.3 of CPE maintains full backward compatibility with version 2.2, but improves upon it by being more rigorous, detailed, and precise, and by adding several new features requested by members of the CPE community. A related ISO "software identification tagging" effort is standardizing a way to tag software products of all kinds.[2] A significant difference between CPE and software ID tags (SWIDs) is that SWIDs are intended to be installed on the computing endpoint at installation, and would require the participation of the software vendors to do this. The CPE team is currently in dialogue with leaders of the SWID effort to explore ways for each effort to leverage and collaborate with the other.

2.1.6 CVSS

Sponsor	Developer	URL
---	FIRST	http://www.first.org/cvss/

The Common Vulnerability Scoring System (CVSS) is a metric to assign a score to software vulnerabilities to help users prioritize risk. The CVSS algorithm is split into three sub-computations: the base metric, the temporal metric, and the environmental metric. The base metric measures inherent characteristics of the vulnerability itself, such as whether it can be exploited remotely, how difficult it is to exploit, whether prior access to the target is required, and its impact on confidentiality, integrity, and availability (CIA). The temporal score reflects aspects of the attack that change with time, such as whether remediations exist, whether there have been instances of exploitation, and the general belief in the accuracy of the vulnerability report. The environmental score attempts to measure characteristics about how the vulnerability will affect a specific enterprise, considering such things as the potential damage, the level of exposure of vulnerable systems, and the enterprise's requirements for CIA. The base metric is required to create a final score, but the other metrics are optional and need only be included if deemed necessary or informative. The final score, regardless of the sub-metrics chosen, ranges between 0 (no concern) and 10 (extreme risk). In addition to the final score, the chosen values for the component characteristics can be presented in a condensed vector. The vector allows a viewer to understand the choices that resulted in the score.

CVSS is not currently sponsored by US government agencies but is relevant to this document both because it is one of the six core SCAP standards and because of the strong influence government agencies played in its creation. CVSS was originally developed as a project of the National Infrastructure Advisory Council (NIAC) within DHS and with significant contributions from NIST. The first version of CVSS was published in early 2005. Shortly thereafter, management of the effort was transitioned to the Forum of Incident Response and Security Teams (FIRST). FIRST oversaw an extensive review of CVSS resulting in the release of version 2 in 2007. FIRST continues to perform advocacy and provide assistance for users of the standard.

The second version of CVSS has increased the community support and acceptance of this standard. Although FIRST's CVSS Special Interest Group (SIG) currently is discussing some changes (e.g., reduce the scoring bias towards the physical host and focus on business-critical applications and data that reside on the host), there is no formal revision scheduled at this time.

[2] See http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=53670.

2.1.7 OCIL

Sponsor	Developer	URL
NSA/NIST	MITRE	http://scap.nist.gov/specifications/ocil/

The Open Checklist Interactive Language (OCIL) is a language to provide a standard way of querying a human user. While focused primarily on serving the needs of security benchmarks, OCIL could easily be applied to broader use cases. It contains structures to encapsulate questions, present procedures for determining answers to questions, evaluate results, and even provide a tree of follow-up questions based on a user's response. The history of OCIL can be traced back to 2005 when the Center for Internet Security (CIS) released a questionnaire language called the "Question Schema" for public use and development, but was unable to support further development on its own. At the same time, MITRE had been developing its own questionnaire language for use in benchmarks but this work had not been published. In 2007, MITRE combined its own work with CIS's Question Schema and published the "Interactive Schema". In 2009 the language was renamed OCIL. The second version of the language has been adopted by the SCAP community and included in the 2011 release of the SCAP 1.1 specification.

Planned development will extend the use case to embrace an Enterprise model. This will move the standard's focus from the current per-machine direct query of a single operator to a policy based administration tool asking questions which would address a number of machines. This will significantly reduce the interaction needed to manage large populations of systems.

2.2 Emerging and Related Standards

This section discusses several security automation standards and suites of standards that cover a variety of use cases and applications, from software assurance to remediation to event management. Some grouping has been provided among standards that are related or are in the same application space for the sake of easier reading. These groupings should not be misinterpreted as a formal umbrella or a strict delineation. For example, EMAP is a formal umbrella for four event standards, while the many of the software assurance-related standards are interrelated but do not have a formal hierarchical structure. Additionally, relationships between standards in different groups exist, as many standards tackle different but related pieces of a security problem.

2.2.1 EMAP

Sponsor	Developer	URL
NIST	NIST/G2/MITRE	http://scap.nist.gov/emap/

The Event Management Automation Protocol (EMAP) is a high-level umbrella for efforts related to standardizing the way in which events are recorded and processed. This effort is intended to be a peer of SCAP, comprising several related security standards. The effort is currently in its early stages and is still in the process of articulating its scope and objectives. A whitepaper is expected out late in 2011. NIST is the sponsor of this work with a partnership between G2, NIST, and MITRE driving development.

While EMAP's focus is on security relevant events, the effort's authors are seeking to produce a framework that is flexible enough to handle all forms of events. A list of possible component standards includes:

- **Common Event Rule Enumeration (CERE)** – An enumeration of vendor-independent rules and filters for matching and event processing;
- **Common Event Expression (CEE)** – A language encapsulating the syntax, taxonomy, and transport of event logs and descriptions;
- **Open Event Expression Language (OEEL)** – A language to encapsulate instructions for translating vendor-specific formats of event information into CEE.
- **Common Event Scoring System (CESS)** – A metric for ranking event severity.

Of the above standards, only CEE currently exists. In addition to these listed standards, it is likely that EMAP will make use of existing standards including CybOX, CAPEC, CWE, CVE, CPE, and CCE.

2.2.1.1 CEE

Sponsor	Developer	URL
NSA	MITRE	http://cee.mitre.org/

As noted above, the Common Event Expression (CEE) is the only component in the EMAP suite that currently exists and CEE is expected to provide a core piece of its functionality. Of note, early in the history of CVE, serious consideration of covering "events" was done under the moniker of the Common Intrusion Event List (CIEL) but was not viable at that time. The current CEE standardizes the way in which all types of computer events are described, logged, and exchanged. The v.0.6 CEE schema was released in July 2011, excluding the Transport Requirements and Transport Syslog Mapping, which were released in October 2011.

The CEE effort covers four components: a taxonomy of event types called the Common Dictionary and Event Expression Taxonomy (CDET), a standardized log syntax called Common Log Syntax (CLS), a transport format for the exchange of events called Common Log Transport (CLT), and a set of best practices for logging, called Common Log Recommendations (CELR). The first three combine to form the CEE "language". The latter item is not a standard, but provides guidelines that not only would help improve the efficacy of a product's logging capabilities, but would also help to make that product's log events more compatible with the CEE language.

The Common Log Transport (CLT) provide features necessary to support the end-to-end audit process by extending the event record representation to include the essential confidentiality, integrity, and availability of audit services. This allows systems to share log information with each other, a repository, or end user in a standard way. A CLT Protocol must meet a given set of tiered requirements, which are based on an enterprise's particular environment. These include core, basic, and optional (optional because these requirements will not be applicable to all environments). For example, a CLT Protocol core requirement is to be able to transmit a CLS Encoded CEE Event. More advanced CLT Protocols may provide things like encryption and full acknowledgments. The CEE CLT component also defines transport mappings. A CLT mapping defines a standardized way for CEE Events to be transmitted over a certain CLT Protocol.

A website update in October 2011 included the following published documents:

- CEE Profile Specification
- Log Syntax Specification
- Profile Repository
- Log Transport Requirements

2.2.2 NVD

Sponsor	Developer	URL
DHS	NIST	http://nvd.nist.gov/home.cfm

The National Vulnerability Database (NVD) is a resource provided by NIST and funded by US-CERT at DHS. NVD serves as the web portal for all NIST's SCAP resources, including libraries of benchmarks, links to the component standards, and support for the validation program. Of primary relevance to this document, NVD also refers to a database of information keyed off the complete list of CVE identifiers. Every CVE identifier is annotated with a CVSS score and vector; a set of references to related solutions, advisories, and tools; a list of CPE entries denoting the affected software, and a CWE entry indicating the nature of the underlying weakness. This list is updated daily as new CVEs are created.

NVD is a follow on to NIST's ICAT "metabase", which served a similar purpose. (The term ICAT was not used as an acronym, although in its initial conception it was intended to be one.) In 2005, NVD was launched as a replacement. NVD has undergone several revisions in format and structure and is currently at version 2.2.

2.2.3 Remediation

Sponsor	Developer	URL
NSA	MITRE	None

Currently the standards contained within SCAP focus on capabilities for the detection, description, scoring, and reporting of flaws, misconfigurations, and attacks. However, to date there has been little standardization of actions to take in response to these vulnerability indicators. Standardizing remediation actions has recently become a topic of significant interest. NIST initiated this effort and in April 2011 published IR-7670: "Proposed Open Specifications for an Enterprise Remediation Automation Framework". This overview maps out several constituent standards:

- **Common Remediation Enumeration (CRE)** – An enumeration where each entry will describe one set of actions one could take in order to address a vulnerability, misconfiguration, or policy violation. Descriptions will be in prose (human-comprehensible vs. machine-comprehensible) format. Because any given vulnerability, etc., might be dealt with in multiple ways, there will likely be multiple CREs associated with any given issue. CRE describes the data that is required to support the technical use cases identified; it does not prescribe a database format, schema or presentation model.
- **CRE Data Exchange Format (CRE-DEF)** – An exchange format for CRE entries and related metadata. This transport format allows the exchange of either the standard CRE list or organization-specific CREs. The CRE data exchange format is envisioned as a lightweight, XML-based schema that serves as the standard import, export, and exchange format for basic remediation information as provided by CRE.
- **Extended Remediation Information (ERI)** – A dictionary with additional data about each CRE. Examples of relevant data could include references to CPEs, CVEs, and CCEs; prerequisites for the remediation action; extended descriptions of the remediation steps; and follow-up actions for both successful and failed attempts to apply the remediation. ERI does not prescribe a database format or schema or any other presentation model. It simply identifies the additional data that may be required to support the identified technical use cases, beyond the base CRE entries.
- **Extended Remediation Data Exchange Format (ERI-DEF)** – The Extended Remediation Information Data Exchange Format is proposed as a means of enabling efficient interchange of ERI data. The ERI data exchange format is envisioned as an XML-based schema that extends the CRE schema, allowing ERI documents to refer to the CRE entries they extend by CRE ID alone, or to contain the full contents of the CRE entry.
- **Remediation Policy Specification (RPS)** The Remediation Policy Specification defines how to associate particular remediations with various classes or types of IT assets. Such a capability allows organizations to specify allowed, preferred, or required remediations for specified collections of IT assets. For example, RPS might filter CREs based on platform type, software inventory, vulnerability presence, configuration states, and functional or organizational categories.
- **Remediation Tasking Language (RTL)** – provides a standardized format to direct compliant tools to enact specific remediations on specific assets. RTL documents

represent the output of the remediation decision process, and function as a standardized input format for remediation tools. Similar in concept to PLARR, RTL would be used to initiate remediations and control where and how those remediations should be performed.

- **Remediation Results** – A language to encapsulate the results of a remediation attempt. Remediation Results convey the outcome (e.g., success/failure/error) of attempted remediation actions as reported by the remediation tool. Remediation Results also enable roll-up reporting and provide enhanced situational awareness.

This initial list of proposed standards and their contents is preliminary and may change as either the overall architecture or each standard is further refined.

2.2.4 Configuration-Related Standards

2.2.4.1 CCI

Sponsor	Developer	URL
DISA	DISA	http://iase.disa.mil/cci/index.html

The Common Configuration Identifier (CCI) effort seeks to create an enumeration of information assurance controls and standards. Specifically, it enumerates high-level policy objectives that are atomic, actionable, and measurable. For example, "enforce minimum password length" would be a high level policy objective appropriate for encapsulating in a CCI. A benchmark recommendation, which specifies values for configuration elements, could be annotated with a CCI ID. This would indicate the high-level policy objective that a given recommendation supports. In this regard, CCI provides a valuable service by mapping low-level recommendations to the high-level objectives they serve. Since many policy statements and auditing requirements are written in terms of high-level objectives, this mapping is useful for providing evidence of organizational audit and policy compliance.

Originally published in 2008, version 2 release 0.1 of the CCI specification was released in February 2011, and the list currently contains over 1600 entries. In addition to the enumeration, the CCI effort includes a proposal for a dictionary that will annotate the CCI list with additional information linking entries to important high-level policy documents, such as those published by the DoD, NIST, and other organizations. Currently, CCIs are created by DISA, which also manages the CCI list and dictionary.

2.2.4.2 CCSS

Sponsor	Developer	URL
NIST	NIST	http://csrc.nist.gov/publications/nistir/ir7502/nistir-7502_CCSS.pdf

The Common Configuration Scoring System (CCSS) is a method for scoring the severity of software security configuration issues. Conceptually, it adds the same value to CCE entries that CVSS adds to CVE entries, but there is a significant difference. Because CCEs do not imply any particular setting for a given configuration control, CCSS only makes sense when considering

some specific setting thereof. That is, while a CCE identifies the configuration control, CCSS applies to one (or more) specific configurations of that control. The first version of CCSS was published in 2008, with a final NIST Interagency Report (IR) published in 2010. CCSS 1.0 may be included in the SCAP 1.2 specification.

The design of CCSS closely follows the design of CVSS. Like CVSS, CCSS is comprised of three component metrics: a base metric, a temporal metric, and an environmental metric. The base metric covers characteristics such as whether a particular configuration will allow unauthorized access (active exploit) versus preventing authorized access (passive exploit), whether the exploitation can be done locally or remotely, whether prior access is required, and what the impact is on CIA (Confidentiality, Integrity, and Availability). The temporal metric considers such concerns as whether there are known exploits taking advantage of a particular configuration and whether measures other than changing the specific configuration are available to mitigate exploits. The environmental metric reflects issues such as whether the given configuration is common within an enterprise, what the value of a vulnerable target is to an attacker, whether mitigations are in place, and what the CIA impact is on an enterprise. As with CVSS, only the base metric is required to produce a score and final scores range from 0 to 10. A vector allows viewers to see the choices that went into the creation of a score.

2.2.4.3 CMSS

Sponsor	Developer	URL
NIST	NIST	http://csrc.nist.gov/publications/drafts/nistir-7517/Draft-NISTIR-7517.pdf

The Common Misuse Scoring System (CMSS) is a metric to rank software feature misuse vulnerabilities. According to the CMSS specification, "a software feature misuse vulnerability is present when the trust assumptions made when designing software features can be abused in a way that violates security." A misuse vulnerability differs from a normal vulnerability (such as one that might warrant a CVE) in that the latter is caused by an error in implementation that allows unintended actions. A misuse vulnerability reflects violations of security made possible by software features that were intentionally included to provide some benefit to the user. The first draft of CMSS was published in early 2009, and the IR is still in Draft form.

CMSS follows the designs of CCSS and CVSS. There are three component metrics, a required base metric and optional temporal and environmental metrics that combined produce a score between 0 and 10. The characteristics considered in the CMSS temporal and environmental metrics are identical to those of CCSS and its base metric. It differs only in that it does not consider active versus passive exploits and instead attempts to provide a value reflecting the complexity of exploiting the misuse vulnerability.

The CMSS specification notes that the concept of "software feature misuse vulnerability" is not as simple as those of a software flaw or a configuration error, as scored by CVSS or CCSS, respectively. To address this confusion, the authors of CMSS have proposed the creation of a dictionary of such vulnerabilities. While the nature of this dictionary is not spelled out in detail, this could imply the need to create and maintain a new enumeration.

2.2.5 Software Assurance-Related Standards

2.2.5.1 CAPEC

Sponsor	Developer	URL
DHS/NSA	MITRE	http://capec.mitre.org/

The Common Attack Pattern Enumeration and Classification (CAPEC) is a dictionary identifying and describing attack patterns and their characteristics. The idea of discussing patterns of attack was originally publicly introduced by Gary McGraw and Greg Huglund in the book "Exploiting Software" in 2004, and has been refined and expanded into today's CAPEC initiative and content which was originally published in 2007. Attack patterns are ways in which attacks on software occur. Buffer overflows, session hijacking, various forms of cross-site scripting, and control injection are just some examples of attack patterns. In addition to an ID and a short description, CAPEC entries can contain such information as the (abstract) steps of a typical exploit; the typical severity and likelihood of exploit; indicators and mitigations; related weaknesses, vulnerabilities, and other CAPECs; and related security and design principles.

In 2011 CAPEC made several significant expansions and revisions. CAPEC now contains 15 attack pattern categories, including four added this year: Network Reconnaissance, Physical Security Attacks, Social Engineering Attacks, and Supply Chain Attacks. These additions expanded CAPEC from solely addressing attacks against software to attacks against systems. CAPEC is currently at version 1.6 and contains 460 identified patterns. Additionally, CAPEC established and then made significant revisions to an Observables sub-schema which is now being pulled out as a separate schema called the Cyber Observables eXpression (CybOX). CybOX will be shared and utilized by CAPEC, MAEC, and CEE.

2.2.5.2 CWE

Sponsor	Developer	URL
DHS	MITRE	http://cwe.mitre.org/

The Common Weakness Enumeration (CWE) is an encyclopedia identifying and describing weaknesses in software architecture, design, implementation, and deployment. CWE traces its history back to MITRE's 2005 Preliminary List Of Vulnerability Examples for Researchers (PLOVER), a collection of 1500 illustrative CVE entries divided up into 290 types. CWE, first released in 2006, focused on the underlying weaknesses themselves and expanded and elaborated on this type list.

The current version of CWE, released in September 2011, is CWE 2.1 which represents a significant milestone since the original release of CWE 1.0 in September 2008. Since then, over 150 new entries have been added. Major changes were also made to 93% of the 734 entries that were in CWE 1.0. More than half of the CWE 1.0 entries had their descriptions or relationships change; more than 30% had changes in their names or demonstrative code examples. The release of CWE 2.0 coincided with the releases of the 2011 CWE/SANS Top 25, CWSS, and CWRAF, all of which also drove improvements to CWE itself.

CWE now includes 886 identified weaknesses and weakness categories. CWE is arranged in a variety of hierarchies with more general weaknesses serving as parents for more specific weaknesses. Entries contain information such as a text description of the weakness, when and

how the weakness is usually introduced, common consequences, code examples, mitigations, known examples in CVEs, and cross references to other CWEs and CAPECs. The CWE web site presents multiple hierarchies or views of this data to aid researchers and developers.

CWE is seeing good penetration with code analysis tools and is generally regarded as the most comprehensive public list of software weaknesses available. Since 2009, the CWE team has been working with SANS to publish the Top 25 Most Dangerous Software Errors[3], which is a list of 25 of the most serious software weaknesses. Recently, the third annual list has received a great deal of national media attention and has proven to be a valuable tool for both educating developers and raising awareness of secure programming practices to a wider community.

2.2.5.3 CWRAF

Sponsor	Developer	URL
DHS	MITRE	http://cwe.mitre.org/cwraf/

The Common Weakness Risk Analysis Framework (CWRAF) provides a means for software developers and consumers to prioritize software weaknesses that are relevant for their business, mission, and deployed technologies. In certain circumstances, a software weakness can lead to an exploitable vulnerability.

By providing a repeatable way to apply the Common Weakness Scoring System (CWSS) to a specific type of application for a specific type of business, CWRAF enables people to reason and communicate about the relative importance of different weaknesses within the context of their business.

Users can automatically generate a more targeted specification of "Top-N" lists of weaknesses that are the most critical for the software that is used in the relevant business domains, missions, and technology groups. In conjunction with other activities, CWRAF ultimately helps developers and consumers to introduce more secure software into their operational environments.

CWRAF includes a mechanism for measuring risk of weaknesses in a way that is closely linked with the risk to the business or mission supports the automatic selection and prioritization of relevant weaknesses.

Customized to the specific needs of the business or mission in conjunction with the Common Weakness Scoring System it can be used by consumers to identify the most important weaknesses for their business domains. CWRAF provides a methodology for applying CWSS to rank classes of weaknesses independently of any particular software package, in order to prioritize them relative to each other (e.g. "buffer overflows are higher priority than memory leaks"). This approach, sometimes referred to as a "Top-N list," is used by the CWE/SANS Top 25 Most Dangerous Software Errors effort. In practice, CWRAF will allow users to create their own custom Top-N lists and CWSS will allow their tools to be driven by that same set of priorities.

[3] http://www.sans.org/top25-software-errors/

2.2.5.4 CWSS

Sponsor	Developer	URL
DHS	MITRE	http://cwe.mitre.org/cwss/index.html

The Common Weakness Scoring System (CWSS) provides a mechanism for scoring software weaknesses and is currently at version 0.8. Modeled after CVSS, this standard provides a method for calculating the score for individual CWEs.

It has been observed that different scanning tools assign different severity scores to software design weaknesses, some assigning one priority while others assign the same flaw another value. In a recent experiment, running a set of such tools against a large open-source project discovered over ten-thousand findings, many that were un-exploitable false-positives. Thus it is clear that researchers need to apply some sort of culling decision to pare the list of findings down to a manageable size. A CWSS metric addresses the inconsistencies between the severity scores that existing tools assign to their findings.

The advantages of CWSS are:

- It provides a common framework for prioritizing security errors ("weaknesses") that are discovered in software applications;
- It provides a quantitative measurement of the unfixed weaknesses that are present within a software application;
- It can be used by developers to prioritize unfixed weaknesses within their own software;
- In conjunction with the Common Weakness Risk Analysis Framework (CWRAF), it can be used by consumers to identify the most important weaknesses for their business domains, in order to inform their acquisition and protection activities as one part of the larger process of achieving software assurance.

2.2.5.5 MAEC

Sponsor	Developer	URL
DHS	MITRE	http://maec.mitre.org/

Malware Attribute Enumeration and Classification (MAEC, pronounced "mike") is a language for characterizing malware based on its attributes, most notably in terms of attack patterns, low-level observables, and behaviors. MAEC is still in the early stages of development, with version 1.1 of the XML schema released in January 2011. The current focus is on the completion of version 2.0 of the schema, which incorporates several major revisions over the previous versions, most notably in terms of incorporating the objects defined in CybOX. Unlike the Common Malware Enumeration (CME), an earlier MITRE effort that is no longer under development, MAEC does not attempt to provide an ID for specific instances of malware but instead seeks to provide a common language with which to describe the characteristics of malware. This matches the broader shift among anti-malware vendors of moving away from enumerative, signature-based detection mechanisms and towards more heuristic-based detections.

MAEC currently consists of an XML schema which defines the language, its objects, and constructs. The current version of the schema, version 1.1, is focused primarily on defining the actions performed by malware at the system level. Such actions can be profiled by instrumenting a system to observe the state changes effected by malware, something typically done through dynamic analysis tools. Examples of such actions include changes to the file system or registry, process modification, and the establishment of network connections. The MAEC schema also defines constructs called behaviors, which serve to group collections of these actions based on their higher order functionality. For example, an email address harvesting behavior can consist of several low-level file searches and read operations that target specific types of files.

The MAEC effort is intended to solve the issues of non-standardized malware reporting, duplication of malware analysis efforts, delayed malware detection, indicator and signature sharing, and others that stem from the lack of a standard for communicating detailed, unambiguous information about malware. MAEC currently links to other standards in order to better characterize elements relevant to malware, including CPE for expressing the platforms targeted by malware and CVE for specifying any vulnerabilities exploited by malware. In addition, a script was developed to take MAEC XML instances, extract relevant objects defined as created by the malware (e.g. files, registry keys) and convert them into OVAL XML, permitting malware detection based on these objects. The next version of MAEC will import and utilize the object types defined in CybOX, establishing compatibility between observables expressed through CEE and CAPEC.

2.2.5.6 CybOX

Sponsor	Developer	URL
DHS	MITRE	Website expected to go live late 2011 http://capec.mitre.org/data/xsd/observables_v0.4.xsd

The Cyber Observable eXpression (CybOX) is a language schema for specifying, capturing and communicating information on Cyber Observables in a consistent, structured and machine consumable manner. The Cyber Observables construct is intended to capture and characterize events or stateful properties that are observable in the operational domain. The concept and initial schema were developed as part of the CAPEC effort in order to annotate specific elements of an attack pattern's detailed attack execution flow in order to describe what that element of the attack might look like in the real world. The initial schema was published as part of CAPEC v1.4 and a significantly updated and enhanced version was included as part of CAPEC v1.6. Shortly after the publication of the initial schema, it was discovered that a need for this concept and construct was not unique to CAPEC but was also relevant to MAEC (for characterizing malware actions), CEE (for modeling events) and other operational information standardization efforts. It is currently fully integrated into CAPEC, partially integrated into MAEC and is in the process of being integrated into CEE. Beyond these efforts, it is a key element in the work of many other parties, including US-CERT, NIST's EMAP effort, and several independent research projects. CybOX has been the topic of presentations at a great many conferences as well as the subject of a recent article in the September/October 2010 issue[4] of Crosstalk magazine.

[4] http://www.crosstalkonline.org/storage/issue-archives/2010/201009/201009-Barnum.pdf

CybOX will provide an open specification to capture observable events or stateful properties to be parsed, filtered, and correlated among other standards. The dedicated website for CybOX is scheduled to go live late in 2011 and will be integrated into the overall set of Making Security Measurable sites.[5]

2.2.5.7 SAFES

Sponsor	Developer	URL
NSA	MITRE	Website expected to go live in late 2011

The Software Assurance Findings Expression Schema (SAFES) is a unified schema that will provide the ability to report, integrate and analyze findings in a consistent fashion. This establishes more structured tool results that are more useful to users, enable integration of results from multiple tools/services and enable automated processing of tool/service results.

This is a collaborative community effort with MITRE providing primary technical leadership but with the involvement and contributions of interested software assurance tool and service vendors and other members of the software assurance community.

The primary artifact for this effort is an XML-based schema that not only provides a common communication mechanism for findings but does so in a structured fashion that enables greater flexibility in its application and its future growth and enhancement. No such comprehensive common schema exists or is openly under development anywhere today; all tool vendors and assessment practitioners utilize their own unique, proprietary schemas. A consistent schema will also aid in development of automated processing of tool findings.

SAFES is currently in early development. Once it reaches an adequate level of capability and stability, SAFES is planned to be transitioned to the Object Management Group (OMG) as a formal standard development effort in their Cyber Ecosystem portfolio.

Initial and current sponsorship for SAFES comes from the NSA Center for Assured Software (CAS). Additional sponsorship is expected soon from DHS. A dedicated website for SAFES is scheduled to go live in late 2011 and will be integrated into the overall set of Making Security Measurable sites.

[5] http://measurablesecurity.mitre.org/

2.2.6 Reporting-Related Standards

2.2.6.1 Enterprise Reporting

Sponsor	Developer	URL
NSA	MITRE/NSA	--

Enterprise Asset Reporting is implemented through several closely related standards: Asset Reporting Format (ARF), Asset Identification (AI), and Assessment Summary Reporting (ASR). In addition, there is also an emerging Report Tasking Language which will enable the definition of task requests. Tasking, which does not yet have a formal name, is still early in community development. More details, including the scope and purpose, will be available as the specification becomes more clearly defined. The Policy Language for Assessment Results Reporting (PLARR) is a legacy control language whose functionality will be replaced by these other standards.

These standards and related specifications are described below.

2.2.6.1.1 ARF

Sponsor	Developer	URL
NSA	MITRE/NSA	http://scap.nist.gov/specifications/arf/

Asset Reporting Format is a data model for expressing the transport format of information about assets and the relationships between assets and reports. Previously known as the Assessment Results Format, this standardized data model and language facilitates the reporting, correlating, and fusing of asset information throughout and between organizations.

ARF is vendor- and technology-neutral, flexible, and is suited for a wide variety of reporting applications. The intent of ARF is to provide a uniform foundation for the expression of reporting results, fostering more widespread application of sound IT management practices. ARF can be used for any type of asset (e.g., buildings, sites, and resources), not just IT assets.

2.2.6.1.2 AI

Sponsor	Developer	URL
NSA	MITRE/NSA	http://scap.nist.gov/specifications/ai/

Asset Identification defines the constructs and methods for representing asset identification information and thus can be leveraged by any other specification where identifying assets is required or beneficial. The scope of Asset Identification is limited to a description of how asset management tools can represent asset identification information when communicating it to other tools. It is out of scope of Asset Identification to recommend which identifiers to use or to require that identification information be collected in a certain way or from a certain place. Higher-level specifications, tools, and organizations that implement Asset Identification, however, are encouraged to make these recommendations or specify these requirements in order to support the particular needs of their use cases.

2.2.6.1.3 ASR

Sponsor	Developer	URL
NSA	MITRE/NSA	--

Asset Summary Reporting is intended to define how to represent summary data. ASR is used to represent aggregate information. It may be used by entities who wish to reduce the volume of reporting data by providing a layer of abstraction. The work is in a preliminary stage of community development. More details will be available as the specification becomes better defined.

2.2.6.1.4 Legacy Reporting Specifications

This section describes three standards efforts closely related to the other Enterprise Reporting efforts. None of these three efforts are under active development today, but because they are sometimes referenced in discussion, and because their names have a potentially confusing overlap with efforts listed above, they are covered here.

ARF 0.41

The Assessment Results Format (ARF) is a language to provide a generalized standard format for assessment results, such as might be created by running a benchmark or a vulnerability scan. While XCCDF and OVAL both have their own result reporting formats, ARF improves and generalizes these formats. ARF also supports the creation of summary results, combining component results from multiple assessments. ARF follows from and replaces MITRE's Common Results Format (CRF) effort. The initial specification of ARF was published in 2009 and is in active use within the Department of Defense (DoD).

(Ref: http://measurablesecurity.mitre.org/incubator/arf/)

ASR 0.41

Assessment Summary Results (ASR) is an open specification that provides a structured language for exchanging summarized assessment results data between assessment tools, asset databases, and other products that manage asset information. It is intended to be used by tools that collect detailed configuration data about IT assets and is in active use within the DoD.

(Ref: http://measurablesecurity.mitre.org/incubator/asr/)

PLARR

The Policy Language for Assessment Results Reporting (PLARR, pronounced "pillar") is a control language for ARF 0.41. It is used to query scanning tools, supply parameters for scans, and to dictate the format of responses. PLARR is explicitly tied to the DoD's Assessment Reporting format and is a legacy language. Although it is presently in use by the DoD, it is not being actively developed further in the SCAP context.

(Ref: http://measurablesecurity.mitre.org/incubator/plarr/)

2.2.6.2 CVRF

Sponsor	Developer	URL
---	ICASI	http://www.icasi.org/cvrf

The Common Vulnerability Reporting Framework (CVRF) 1.0 is an XML based language designed to provide a documentation standard for security related information.

Recently there has been significant progress in information system vulnerability categorization and severity ranking. However there is a gap in that documentation of these vulnerabilities occurs in an ad-hoc, vender specific and non-standard format.

Existing security documentation contains similar information (e.g., date fields, overview-type fields, impact, and remediation fields) but there is no common format or nomenclature consistent across the various reports. The proposed solution, the Common Vulnerability Reporting Framework (CVRF) is an XML-based framework that predefines a large number of fields designed with extensibility and robustness in mind. The CVRF project has been initiated by the Industry Consortium for the Advancement of Security on the Internet (ICASI), a vendor neutral industry-wide think tank.

The ICASI CVRF working group has assembled experts to establish a core team that could expand existing security documentation formats and subsequently integrate a best-of-breed solution into a common XML-based framework. An initial white paper was published in May, 2011.

2.2.7 Related International Standards

2.2.7.1 IODEF

Sponsor	Developer	URL
IETF	IETF	http://xml.coverpages.org/iodef.html http://www.ietf.org/rfc/rfc5070.txt

The Incident Object Description Exchange Format (IODEF) is a proposed Internet Engineering Task Force (IETF) standard. It defines a defines a data representation that provides a framework for sharing information commonly exchanged by Computer Security Incident Response Teams (CSIRTs) or Computer Emergency Readiness Teams (CERTs) about computer security incidents. IODEF is focused on human-to-human communication involving incident response. IODEF has the ability to reference other standards (CEE, for example), to provide information related to an incident.

IODEF is currently used by many CERTs for sharing incident information, including the DHS NCSD (National Cyber Security Division) US-CERT. IODEF efforts have now given way to a new IETF effort called Managed Incident Lightweight Exchange (MILE – https://www.ietf.org/mailman/listinfo/mile), which is being followed by members of the MITRE standards team.

2.2.7.2 CYBEX

Sponsor	Developer	URL
ITU-T	ITU-T	http://ccr.sigcomm.org/online/files/p59-3v40n5i-takahashi3A.pdf http://www.itu.int/itu-t/workprog/wp_item.aspx?isn=7305

The Cybersecurity Information Exchange (CYBEX) is a pre-published International Telecommunications Union (ITU-T) standard (X.1500). It was a proposed standard (an ITU-T recommendation) that was accepted in April 2011, and the final specification is currently being published. CYBEX is intended to describe ways in which cybersecurity entities can exchange assured cybersecurity information. The parties exchanging such information will often be CERTs and private companies developing software or network-based systems.

CYBEX references and advocates the use of many of the standards mentioned in this paper, including CVE, CPE, CCE, CAPEC, CWE, MAEC, CEE, OVAL, ARF, CWSS, CVSS, XCCDF, and IODEF. These standards are only references from CYBEX; modification of each is still through the core standard itself. DHS is involved in writing the CYBEX specification. Due to CYBEX's referencing each of the SCAP standards, USGCB (United States Government Configuration Baseline) compliance becomes a use case for CYBEX.

IODEF and CYBEX could be used to move cybersecurity information around as necessary to support incident management amongst the organizations involved in its various aspects and stages.

3 Validation and Adoption Programs

In the early 2000s as CVE and OVAL were gaining community acceptance, some government agencies began putting language in acquisition documents that required the support of these standards. This necessitated the development of formal method of measuring compliance with these standards. To meet this need, MITRE began compatibility programs for both CVE and OVAL. MITRE was able to support this activity at a relatively low level of funding for several years, and discovered that it promoted the adoption of the standards. MITRE's Compatibility Program provided users and vendors with certainty that a product that successfully passed compatibility testing would be interoperable with other products that had passed interoperability testing.

When NIST developed SCAP, they discussed standards functional testing with MITRE and the two organizations agreed that MITRE's Compatibility Program would be discontinued for OVAL and NIST would begin a Validation Program to replace it, although MITRE would continue to oversee compatibility with CVE. In the NIST SCAP Validation Program, the testing is conducted by independent labs which have been accredited by the NIST National Voluntary Laboratory Accreditation Program (NVLAP). NIST has documented the test requirements in NIST IR 7511, *Security Content Automation Protocol (SCAP) Version 1.0 Validation Program Test Requirements*. This document is undergoing its second revision, which was made available for public comment in January 2011 and lays out the testing requirements for each of the SCAP

standards. Furthermore, requirements for independent laboratories to conduct SCAP Validation are defined in NIST Handbook 150, and NIST Handbook 150-17.

Complete details on The SCAP Validation Program can be found on the NIST SCAP site: http://scap.nist.gov/validation/.

4 Conclusions

This document has provided a brief overview of the security automation space. In particular we have included an overview of each of the components that make up this space, both in terms of individual standards as well as the forces that guide the standards.

In over 10 years security automation has grown from a single standard with a handful of participants, to a large community and that is growing every year. The more mature of the standards have been incorporated into hundreds of tools and CVE has become virtually ubiquitous in its subject area. Given the ever-increasing community of adopters, implementers, and contributors, it is clear that the overall security automation effort has been highly successful thus far, and its capabilities and interest in those capabilities continue to grow.

Appendix A Acronym Glossary

AI	Artificial Intelligence
ARF	Asset Reporting Format OR Assessment Results Format
ASR	Asset Summary Reporting OR Assessment Summary Results
CAPEC	Common Attack Pattern Enumeration and Classification
CCE	Common Configuration Enumeration
CCI	Common Configuration Identifier
CCSS	Common Configuration Scoring System
CEE	Common Event Expression
CDET	Common Dictionary and Event Expression Taxonomy
CELR	Common Log Recommendations
CERE	Common Event Rule Enumeration
CERT	Computer Emergency Readiness Team
CESS	Common Event Scoring System
CIA	Confidentiality, Integrity, and Availability
CIRT	Computer Incident Response Team
CLS	Common Log Syntax
CLT	Common Log Transport
CME	Common Malware Enumeration
CMSS	Common Misuse Scoring System
CPE	Common Platform Enumeration
CRE	Common Remediation Enumeration
CRE-DEF	CRE Data Exchange Format
CSIRT	Computer Security Incident Response Team
CVE	Common Vulnerabilities and Exposures
CVRF	Common Vulnerability Reporting Framework
CVSS	Common Vulnerability Scoring System
CWE	Common Weakness Enumeration
CWRAF	Common Weakness Risk Analysis Framework
CWSS	Common Weakness Scoring System
CybOX	Cyber Observable eXpression

CYBEX	Cybersecurity Information Exchange
DHS	Department of Homeland Security
DISA	Defense Information Systems Agency
DoD	Department of Defense
EMAP	Event Management Automation Protocol
ERI	Extended Remediation Language
ERI-DEF	ERI Data Exchange Format
FIRST	Forum of Incident Response and Security Teams
GPO	Group Policy Object
ICASI	Industry Consortium for the Advancement of Security on the Internet
ICAT	ICAT (not an acronym)
IODEF	Incident Object Description Exchange Format
ISS	Internet Security Systems
MAEC	Malware Attribute Enumeration and Classification
NIAC	National Infrastructure Advisory Council
NIST	National Institute of Standards and Technology
NIST IR	NIST Interagency Report
NIST NVLAP	NIST National Voluntary Laboratory Accreditation Program
NSA	National Security Agency
NSA CAS	NSA Center for Assured Software
NVD	National Vulnerability Database
OCIL	Open Checklist Interactive Language
OEEL	Open Event Expression Language
OMG	Object Management Group
OVAL	Open Vulnerability and Assessment Language
OVALDI	OVAL Definition Interpreter
PLARR	Policy Language for Assessment Results Reporting
RPM	RPM Package Manager[6]
RPS	Remediation Policy Specification
RTL	Remediation Tasking Language
SAFES	Software Assurance Findings Expression Schema

[6] RPM is a recursive acronym. It once stood for “Red Hat Package Manager”.

SANS	SysAdmin, Audit, Networking, and Security
SCAP	Security Content Automation Protocol
SIG	Special Interest Group
SQL	Structured Query Language
SWID	Software Identification
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
US-CERT	United States Computer Emergency Readiness Team
USGCB	United States Government Configuration Baseline
XCCDF	eXtensible Configuration Checklist Description Format
XML	eXtensible Markup Language