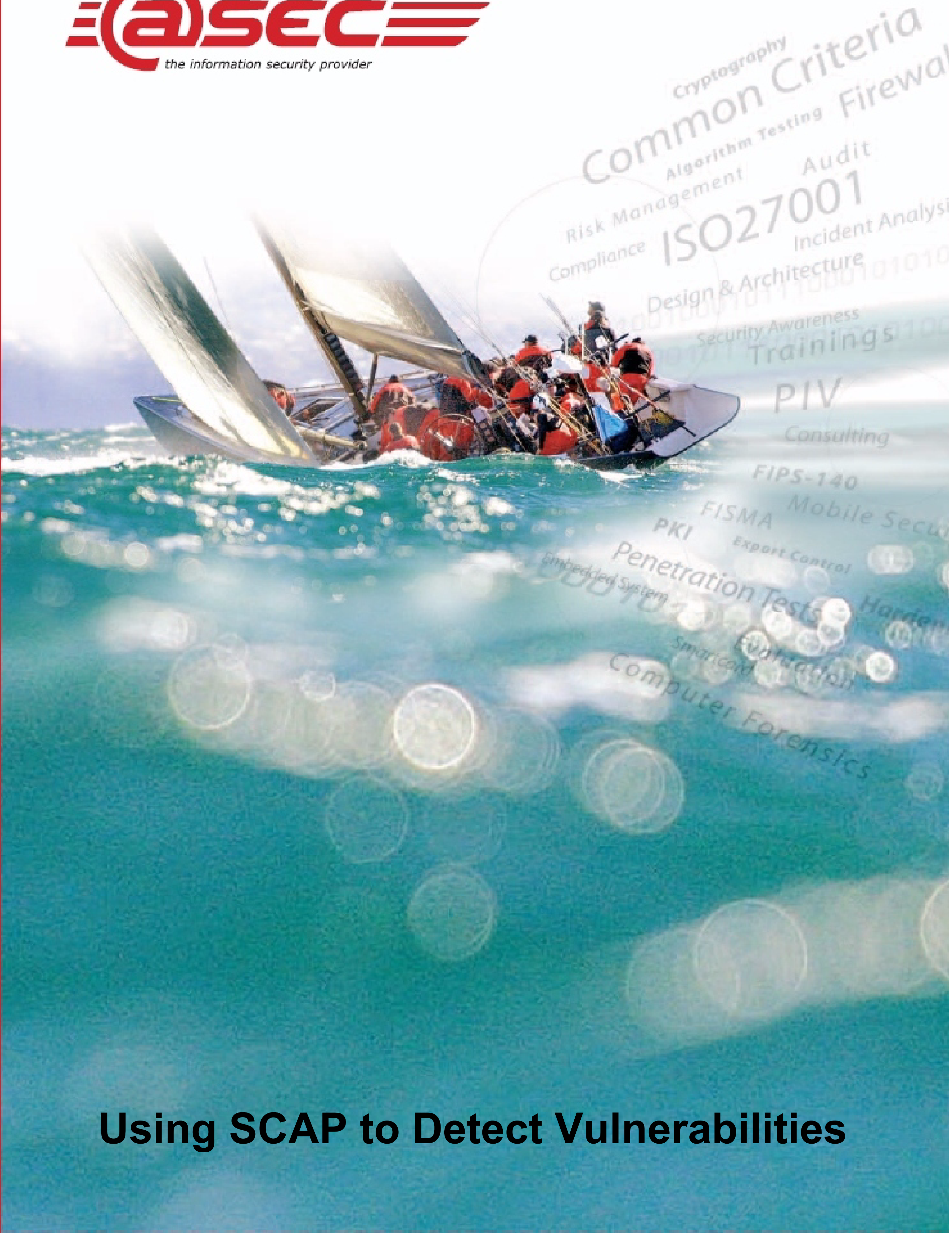
the information security provider



Using SCAP to Detect Vulnerabilities

Using SCAP to Detect Vulnerabilities

by Steve H. Weingart

April 2008

Introduction

Security Content Automation Protocol (SCAP), pronounced "S" "CAP", is an effort by the U.S. National Institute for Standards and Technology (NIST) to standardize the way security vulnerability and configuration information is identified and cataloged. The advantage of this method is that each item is identified once - the common scheme prevents multiple entries for a single issue. It is identified in one place where it is easily found and the list is available at no cost to anyone who wants it, so everyone can work from the same information.

SCAP started with six standards (called 'components') that enumerate different types of vulnerabilities and exposures [1]. A "vulnerability," or "flaw," is a mistake in software that can be directly used to gain access to a system or network. An "exposure" is a system configuration issue or a mistake in software that allows access to information or privileges that can be used as a path into a system or network.

Each component addresses vulnerabilities and exposures from a different perspective. One component is a standard list of known vulnerabilities, while another lists system configuration items so a configuration can be completely and explicitly defined and managed; another component is used to identify operating systems, applications and packages in a uniform manner, so that you can uniquely identify a particular version.

The components are useful individually, but the real value is when several components are combined to identify a system, specify a configuration and then identify exposures or vulnerabilities. A combination of SCAP components used for a particular purpose is said to have SCAP 'capabilities.' Applications that have capabilities and that can be tested and validated to the SCAP standard currently include: scanning tools for configuration and vulnerabilities, remediation tools, and asset management tools (asset, vulnerability and misconfiguration database tools and malware tools).

Each of these tools is a combination of SCAP components and the SCAP standard identifies a set of conformance tests that the applications must pass to be validated under the SCAP standard. More tools will be added as needs are identified and the capabilities are defined.

In addition, since the SCAP data feeds are all completely defined, and some of them are extensible (such as the checklist tools), you can build your own tools using SCAP data as well as using commercial, open source or freeware tools to examine or manage your systems.

Before discussing how to use the components, the following sections describe each of the component standards and tell how they can be used to help manage the security of a system.

SCAP Components

Here, each of the six SCAP components is explained and its sources and methods identified. Much of this information comes from the website for each of these standards; please see these websites for complete information.

CVE

CVE is a system for listing known vulnerabilities. When a new vulnerability is found, it is submitted and verified before it is added to the list. The listing includes a unique identification as well as a description and references about the vulnerability, what is the weakness and how it is applied [2].

When someone discovers what they believe is a new vulnerability, they can go to the Candidate Numbering Authority to request a new CVE ID. The candidate is proposed to the CVE Review Board and if it is verified and determined to be a new CVE, it is assigned a new CVE ID, if it turns out to be a duplicate, or it cannot be verified, then no new ID is assigned.

The policies used to determine the validity of a new CVE are the criteria and consistency rules that determine which security issues become CVE candidates for inclusion in the CVE list, and how the reviewers distinguish between similar or related security issues.

Typically, the approach is to create separate candidates for vulnerabilities of different types, vulnerabilities of the same type that appear in different versions and vulnerabilities that are found in different codebases (this could be by vendor; or it could group vendors who share the same code such as Linux/Unix vendors).

If you want to examine the CVE list there are several ways, you can go to the CVE list at http://cve.mitre.org and view or download a copy of the list. If you want to search the list for specific words, platforms or applications, you can go to the National Vulnerability Database website at http://nvd.nist.gov and search the list.

CCE

Like CVE in the previous section, CCE uniquely identifies important security parameters. However instead of identifying vulnerabilities, CCE identifies configuration elements in operating systems, programs and packages so that there can be common configuration data across multiple sources and tools.

Each entry in the CCE List contains the following attributes: CCE Identifier Number, a human readable description of the configuration item, conceptual parameters (the parameters that would need to be specified in order to implement that CCE), associated technical mechanisms (a particular configuration issue may have one or more methods of resolution) and citations (references, documents or tools in which the item is detailed).

Currently, CCE is focused only on software-based configurations. Recommendations for hardware and/or physical configurations are not supported. [3]

There are CCE lists currently available for: Windows Vista, Windows XP, Windows 2000, Windows Server 2003, Microsoft Office 2007, Internet Explorer, Red Hat Enterprise Linux 5 and Sun Solaris 10.

Each of these lists provides information for virtually every configurable aspect of each of these software items with respect to security.

The lists are valuable in themselves as a catalog of all configuration settings. CCE scanning tools can check and verify each setting to completely map a system or verify it's conformance to a configuration checklist

CPE

CPE is a naming mechanism for systems, platforms, and packages and is based on the syntax for Uniform Resource Identifiers (URI). CPE includes a formal name format, a language for describing complex platforms, a method to check names against a system, and a description format for connecting text and tests to a name. [4]

CPE sets the basis for identification. Before you can manage something as specific as a system's configuration or vulnerabilities, you have to make sure that you are applying the settings to the right system. The purpose for CPE is to explicitly define and enumerate each platform or application, and its version, so that configuration management and vulnerability testing can be performed on exactly the right thing.

Like the other SCAP component standards, CPE has a publicly available data stream. If you have an OS, package or application that is not in CPE, there is a mechanism to get it added to the dictionary.

CVSS

"The Common Vulnerability Scoring System (CVSS) provides an open framework for communicating the characteristics and impacts of IT vulnerabilities. Its quantitative model ensures repeatable accurate measurement while enabling users to see the underlying vulnerability characteristics that were used to generate the scores. Thus, CVSS is well suited as a standard measurement system for industries, organizations, and governments that need accurate and consistent vulnerability impact scores. Two common uses of CVSS are prioritization of vulnerability remediation activities and in calculating the severity of vulnerabilities discovered on one's systems. The National Vulnerability Database (NVD) provides CVSS scores for almost all known vulnerabilities." [5]

CVSS is a flexible system for identifying and objectively quantifying vulnerabilities and their potential impact. The system is made more useful by the different elements of the equation. First a base score is determined. The base score looks at exploitability and potential impact. But that is not enough to really understand the potential damage a particular vulnerability could cause, so there are additional parts to the vulnerability equation. First there is a temporal aspect; this part adjusts the score based on the danger of the vulnerability with respect to time. An exploit that is not completely verified and easy gets a higher score than one that is verified and simple to fix. Lastly there is an environmental factor. A vulnerability that exists on an isolated secure network is less of a threat than one on a publicly accessible one.

Below are the equations for the CVSS Base, Temporal and Environmental scores. The scoring takes into account aspects of the vulnerability and weights them for significance. The potential impact of the flaw, how difficult or complex it is to exploit and what level of access is needed to perform the exploit, as well as other relevant factors, are taken into account and considered to produce the base score.

As can be seen from in equations, weighting is applied to the various elements to produce a score that realistically defines the threat. Additionally, the base score can be modified by the temporal and environmental parts to truly map the vulnerability onto the real conditions being faced.

Additionally, to truly map the vulnerability onto the real conditions being faced, the base score can be modified by the temporal and environmental parts.

CVSS Base Score Equation

BaseScore =
(.6*Impact +.4*Exploitability-1.5)*f(Impact)

Impact =
10.41*(1-(1-ConfImpact)(1-IntegImpact)*(1-AvailImpact))

Exploitability =
20*AccessComplexity*Authentication*AccessVector

f(Impact) =
0 if Impact=0; 1.176 otherwise

AccessComplexity = case AccessComplexity of
- high: 0.35
- medium: 0.61
- low: 0.71

Authentication = case Authentication of
- Requires no authentication: 0.704
- Requires single instance of authentication: 0.56
- Requires multiple instances of authentication: 0.45

AccessVector = case AccessVector of
- Requires local access: .395
- Local Network accessible: .646
- Network accessible: 1

ConfImpact = case ConfidentialityImpact of
- none: 0
- partial: 0.275
- complete: 0.660

IntegImpact = case IntegrityImpact of
- none: 0
- partial: 0.275
- complete: 0.660

AvailImpact = case AvailabilityImpact of
- none: 0
- partial: 0.275
- complete: 0.660

CVSS Temporal Equation

TemporalScore =
BaseScore*Exploitability*RemediationLevel*ReportConfidence

Exploitability = case Exploitability of
- unproven: 0.85
- proof-of-concept: 0.9
- functional: 0.95
- high: 1.00
- not defined: 1.00

RemediationLevel = case RemediationLevel of
- official-fix: 0.87
- temporary-fix: 0.90
- workaround: 0.95
- unavailable: 1.00
- not defined: 1.00

ReportConfidence = case ReportConfidence of
- unconfirmed: 0.90
- uncorroborated: 0.95
- confirmed: 1.00
- not defined: 1.00

CVSS Environmental Equation

EnvironmentalScore =
(AdjustedTemporal+(10-AdjustedTemporal)*CollateralDamagePotential) * TargetDistribution

AdjustedTemporal =
TemporalScore recomputed with the Impact sub-equation replaced with the following AdjustedImpact equation.

AdjustedImpact =
Min(10, 10.41*(1-(1-ConfImpact*ConfReq)*(1-IntegImpact*IntegReq)*(1-AvailImpact*AvailReq)))

```
CollateralDamagePotential = case
CollateralDamagePotential of
    none: 0
    low:  0.1
    low-medium: 0.3
    medium-high: 0.4
    high: 0.5
    not defined:  0

TargetDistribution  = case TargetDistribution
of
    none:  0
    low:  0.25
    medium:  0.75
    high:  1.00
    not defined: 1.00

ConfReq  = case ConfidentialityImpact of
    low:  0.5
    medium:  1
    high:  1.51
    not defined:  1

IntegReq = case IntegrityImpact of
    low:  0.5
    medium:  1
    high:  1.51
    not defined:  1

AvailReq = case AvailabilityImpact of
    low:  0.5
    medium: 1
    high:  1.51
    not defined: 1
```

XCCDF

"XCCDF (Extensible Configuration Checklist Description Format) is a specification language for writing security checklists, benchmarks, and related kinds of documents. An XCCDF document represents a structured collection of security configuration rules for some set of target systems. The specification is designed to support information interchange, document generation, organizational and situational tailoring, automated compliance testing, and compliance scoring. The specification also defines a data model and format for storing results of benchmark compliance testing. The intent of XCCDF is to provide a uniform foundation for expression of security checklists, benchmarks, and other configuration guidance, and thereby foster more widespread application of good security practices". [6]

"XCCDF documents are written in XML and the XCCDF schema can be used with a XML validating parser to validate XCCDF documents".

XCCDF has grown since its inception as a way to define checklists with rules that could be grouped and conditionally executed, to a more general format that supports document generation including user manuals and guides. XCCDF has shown that it is quite flexible and can be extended to many uses. However, its primary function as stated above is to provide configuration checklist documents to meet US Federal standards.

XCCDF was designed to be compatible with different kinds of checking engines. In general practice OVAL based engines are the default.

XCCDF and Oval (below) are the most complex of the SCAP components and for further information on how the XCCDF language is structured, see the XCCDF specification at: http://nvd.nist.gov/scap/xccdf/docs/xccdf-spec-1_1_4-20071011.pdf.

OVAL

"Open Vulnerability and Assessment Language (OVAL) is an international, information security, community standard to promote open and publicly available security content, and to standardize the transfer of this information across the entire spectrum of security tools and services. OVAL includes a language used to encode system details, and an assortment of content repositories held throughout the community. The language standardizes the three main steps of the assessment process: representing configuration information of systems for testing; analyzing the system for the presence of the specified machine state (vulnerability, configuration, patch state, etc.); and reporting the results of this assessment. The repositories are collections of publicly available and open content that utilize the language.

The OVAL community has developed three Extensible Markup Language (XML) schemas to serve as the framework and vocabulary of OVA. These schemas correspond to the three steps of the assessment process:" [7]

- "OVAL System Characteristics schema for representing system information.
- "OVAL Definition schema for expressing a specific machine state.
- "OVAL Results schema for reporting the results of an assessment.

OVAL is probably the most mature of the SCAP components and has a large community of supporters. There is also a large repository of OVAL documents available for download at the OVAL website, for all of the uses that OVAL supports.

Using SCAP Components

The six SCAP component standards are a collection of information that can be used for any number of things. First we can look at just the information that the components provide.

CVE is a list of known vulnerabilities, the list is arranged by platform and can be used to search for vulnerabilities by subsystem (network, service, etc.) or by what the vulnerability provides (stop system, allow access, allow administration). Here is a typical CVE entry.

```
==================================
Name: CVE-1999-0021
Status: Entry
Reference: BUGTRAQ:19971010 Security
flaw in Count.cgi (wwwcount)
Reference: CERT:CA-97.24.Count_cgi
Reference: XF:http-cgi-count
Reference: BID:128
Reference: URL:http://www.securityfo-
cus.com/bid/128

Arbitrary command execution via buffer over-
flow in Count.cgi
(wwwcount) cgi-bin program.
==================================
```

This entry tells what program has the vulnerability, gives a description of what the vulnerability is (a classic buffer overflow) and several references to places that you can get additional information about the vulnerability. All of the vulnerabilities have been checked and verified, so you can be assured that the information is valid.

CCE is a wealth of knowledge of configuration data. Virtually every configurable security parameter for the applications and platforms supported is listed and the preferred setting defined. Currently, there are lists for Windows 2000, XP, Vista and Server 2003; Red Hat EL 5, Sun Solaris 10; Microsoft Office 2007 and Internet Explorer 7. The lists are available for download at the CCE website.

CPE is the component that has the least standalone use. It is valuable for identifying specific platforms and applications by version. Vulnerabilities that are version specific can be pinpointed using CPE for identification.

CVSS can be used to explore vulnerability space to understand where best to apply remediation or mitigation efforts. Clearly, the more serious threats are the ones that need looking at first. In some cases the temporal and environmental modifiers may make the difference between a vulnerability that can be ignored for a time and one that needs immediate response, the reverse could be true as well.

XCCDF itself does not present information directly, but there are many checklists available for download and the rules and information that can be found in these documents can be useful. However XCCDF will prove its greatest value when being used as the definition language for producing checklists used to scan and test systems. Much of the information will be found in the OVAL repository as an OVAL engine is typically used to process an XCCDF document.

OVAL may be the most information rich source of all the SCAP components since the OVAL repository on the OVAL website contains data files of all vulnerability, compliance, inventory, and patch definitions for supported platforms. Virtually all the information available for any supported platform can be found in one place. An OVAL interpreter which can process this data is also available for download from the website. The interpreter is a reference implementation that can interpret definitions and generate output, while not a complete scanning tool, it will provide OVAL IDs and references. The source code and documentation for the interpreter is available for download at the OVAL site on Sourceforge.

SCAP Capabilities

When SCAP components are combined to create more complex and functional tools, these tools are said to have SCAP capabilities. An example of a tool with an SCAP capability is an Authenticated Configuration Scanner, a configuration scanning tool that runs on the system under test with privileges to test and verify the configuration of the system (an FDCC or Federal Desktop Core Configuration scanner is an Authenticated Configuration Scanner).

An Authenticated Configuration Scanner has parts and attributes of a CVE tool, a CCE tool, an XCCDF tool, and an OVAL tool. Under SCAP these features are combined to create an architecture that is able to map requirements, test and report on a system with respect to its configuration. And FDCC Scanner adds CVE and specific FDCC requirements to meet the US Federal desktop configuration scanning requirements.

Currently there are SCAP capability configurations for Authenticated Vulnerability and Patch Scanners, Unauthenticated Vulnerability Scanners, IDPS systems, Patch Remediation systems, Misconfiguration Remediation systems, Asset Scanners; Asset Vulnerability and Misconfiguration databases and Malware tools. The Security Content Automation Protocol (SCAP) Validation Program Test Requirements document on the SCAP website has the complete requirements list. More capability based tools may be defined if needed.

SCAP and SCAP Component-Based Tools

As of March 20, 2008 there are eleven SCAP based tools that have been validated to the SCAP standard by NIST. If you would like to try one of these tools, Threat Guard has released Secutor Prime, an FDCC scanning tool for Windows XP, XP Pro and Vista that is free for personal use. If you search the web for 'SAP compliant' you will find many more tools that claim compliance or are in the process of getting validated by NIST.

If you go to the websites for the SCAP components you will also find references and links to tools that implement one of more of the SCAP components.

SCAP Validation and Testing

With any standard it is critical to be sure that the tools you are using actually implement the standard. It has become common practice over time for some to claim compatibility with a standard while never actually going through independent testing to ensure that the program or tool actually implements that standard correctly.

Such independent testing is managed through NIST's SCAP program [1], and it has the advantage of combining the standards for the six components into itself. This allows the maker of a tool that implements any or all of the SCAP components or capabilities to go through the testing process once. The test labs that are accredited to test for SCAP and to submit results to NIST can make sure that everything works as claimed and that the tool is useful in the way it is described.

Conclusion

SCAP is a valuable new addition to the security world. By combining several standards enumerating security vulnerabilities and flaws, and taking advantage of the combination, all of the critical information can be found in one place. For people trying to understand those flaws and vulnerabilities the data available on the websites for download is a very good reference of known and verified information about configuration and weaknesses. For those trying to protect systems and networks, the information represents the same quality reference, and the available tools, which have been tested and validated gives a strong level of assurance that the tools perform as claimed.

While the SCAP standard was created primarily to meet a US Federal mandate to protect US Governmental computer systems, it will be useful for a much larger population of people who would like to know more about configuration and vulnerability and those who would like to protect their computer systems.

Bibliography

[1] *The Information Security Automation Program and The Security Content Automation Protocol*, http://nvd.nist.gov/scap.cfm

[2] *Common Vulnerabilities and Exposures*, http://cve.mitre.org/

[3] *Common Configuration Enumeration*, http://cce.mitre.org/

[4] *Common Platform Enumeration*, http://cpe.mitre.org/

[5] *Common Vulnerability Scoring System*, http://nvd.nist.gov/cvss.cfm?version=2

[6] *Extensible Configuration Checklist Description Format*, http://nvd.nist.gov/xccdf.cfm

[7] *Open Vulnerability and Assessment Language*, http://oval.mitre.org/