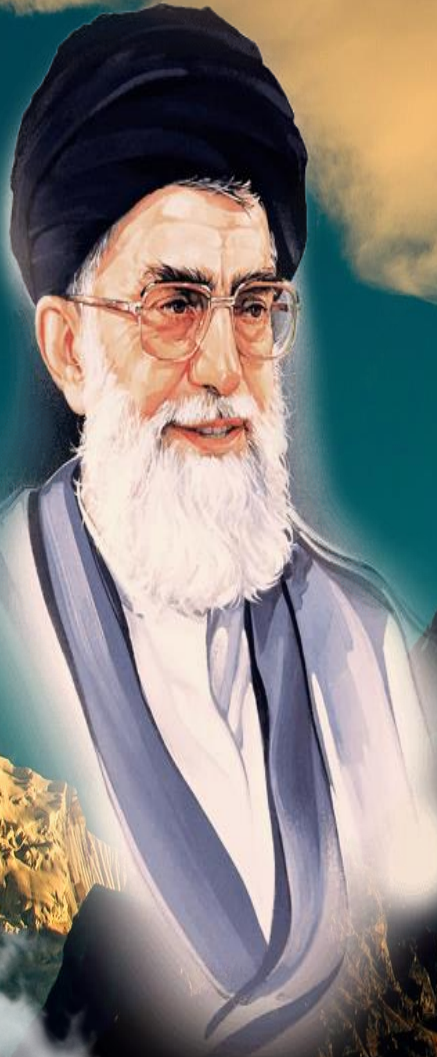




بِسْمِ اللّٰهِ الرَّحْمٰنِ الرَّحِیْمِ





انتخاب ملت ایران در مقابل جبهه استکبار «ایستادگی، عدم تسلیم و وابستگی، حفظ استقلال و تقویت درونی نظام و کشور» است.

برای مشرف عادلانه و عمل مثل فقر حرکت به سمت اقتصاد دانش بنیان باید کرد.

مادر جنگ به این نتیجه رسیدیم که باید روی پای خودمان بایستیم

Security Content Automation Protocol (SCAP)

مشخصات خصوصیات سرمایه های سایبری
Common Platform Enumeration
(CPE) Specifications

Security Content Automation Protocol (SCAP)

- Asset Identification
- Asset Reporting Format (ARF)
- Common Configuration Enumeration (CCE)
- Common Platform Enumeration (CPE)
 - Applicability
 - Language
 - Dictionary
 - Name Matching
 - Naming
- Open Vulnerability Assessment Language (OVAL)
- Open Checklist Interactive Language (OCIL)
- Trust Model for Security Automation Data (TMSAD)
- Extensible Configuration Checklist Description Format (XCCDF)
- Software Identification (SWID)

Common Platform Enumeration (CPE)

- *Common Platform Enumeration (CPE)* is a standardized method of describing and identifying classes of applications, operating systems, and hardware devices present among an enterprise's computing assets. CPE does not identify unique instantiations of products on systems, such as the installation of XYZ Visualizer Enterprise Suite 4.2.3 with serial number Q472B987P113. Rather, CPE identifies abstract classes of products, such as XYZ Visualizer Enterprise Suite 4.2.3, XYZ Visualizer Enterprise Suite (all versions), or XYZ Visualizer (all variations).
- IT management tools can collect information about installed products, identifying these products using their CPE names, and then use this standardized information to help make fully or partially automated decisions regarding the assets. For example, identifying the presence of XYZ Visualizer Enterprise Suite could trigger a vulnerability management tool to check the system for known vulnerabilities in the software, and also trigger a configuration management tool to verify that the software is configured securely in accordance with the organization's policies. This example illustrates how CPE names can be used as a standardized source of information for enforcing and verifying IT management policies across tools.
- The current version of CPE is 2.3. CPE 2.3 is defined through a set of specifications in a stack-based model, where capabilities are based on simpler, more narrowly defined elements that are specified lower in the stack. This design opens opportunities for innovation, as novel capabilities can be defined by combining only the needed elements, and the impacts of change can be better compartmentalized and managed.
- This graphic shows the current CPE 2.3 stack, with the most fundamental layer (Naming) at the bottom. Each higher layer builds on top of the layers below it.

CPE 2.3 stack

Applicability Language

Dictionary

Name Matching

Naming

CPE 2.3 stack

Naming

The Naming specification defines the logical structure of Well-formed Names (WFNs), URI bindings, and formatted string bindings, and the procedures for converting WFNs to and from the bindings.

Name Matching

The Name Matching specification defines the procedures for comparing WFNs to each other so as to determine whether they refer to some or all of the same products.

Dictionary

The Dictionary specification defines the concept of a CPE dictionary, which is a repository of CPE names and metadata, with each name identifying a single class of IT product. The Dictionary specification defines processes for using the dictionary, such as how to search for a particular CPE name or look for dictionary entries that belong to a broader product class. Also, the Dictionary specification outlines all the rules that dictionary maintainers must follow when creating new dictionary entries and updating existing entries.

Applicability Language

The Applicability Language specification defines a standardized structure for forming complex logical expressions out of WFNs. These expressions, also known as applicability statements, are used to tag checklists, policies, guidance, and other documents with information about the product(s) to which the documents apply. For example, a security checklist for Mozilla Firefox 3.6 running on Microsoft Windows Vista could be tagged with a single applicability statement that ensures only systems with both Mozilla Firefox 3.6 and Microsoft Windows Vista will have the security checklist applied.

CPE 2.3 stack

CPE Version 2.3 was released by the [U.S. National Institute of Standards and Technology \(NIST\)](#) in August 2011. CPE v2.3 is also included in Version 1.2 of NIST's [Security Content Automation Protocol \(SCAP\)](#) standard. CPE v2.3 supersedes CPE v2.2, which was released in March 2009. (NOTE: NIST continues to provide support for a CPE v2.2-conformant Dictionary.)

CPE Version 2.3 departs significantly from past CPE practice by breaking the CPE standard into a suite of separate specifications organized in a stack: Naming, Name Matching, Dictionary, and Language. Naming is the foundation of the stack, with each specification building on those that precede it.

Naming

- The CPE 2.3 Naming Specification defines standardized methods for assigning names to IT product classes. An example is the following name representing Microsoft Internet Explorer 8.0.6001 Beta:
 - `wfn:[part="a",vendor="microsoft",product="internet_explorer",version="8\.o\.6001",update="beta"]`
- This method of naming is known as a well-formed CPE name (WFN). It is an abstract logical construction. The CPE Naming Specification defines procedures for binding WFNs to machine-readable encodings, as well as unbinding those encodings back to WFNs. One of the bindings, called a [Uniform Resource Identifier \(URI\)](#) binding, is included in CPE 2.3 for backward compatibility with CPE 2.2 (see the [CPE Archive](#)). The URI binding representation of the WFN above is:
 - `cpe:/a:microsoft:internet_explorer:8.0.6001:beta`

Naming

- The [Official CPE Dictionary](#) published and maintained by NIST contains an authoritative enumeration of CPE names in the URI binding representation.
- The second binding defined in CPE 2.3 is called a formatted string binding. It has a somewhat different syntax than the URI binding, and it also supports additional product attributes. With the formatted string binding, the WFN above can be represented by the following:
 - `cpe:2.3:a:microsoft:internet_explorer:8.0.6001:beta:*:*:*:*:*`
- The WFN concept and the bindings defined by the CPE Naming specification are the fundamental building blocks at the core of all CPE functionality.
- **CPE 2.3 Naming Specification Document and CPE Reference Implementation**
- Go to the [Downloads](#) section below to download the entire CPE 2.3 Naming Specification document, NIST IR 7695. Also available is zip file of MITRE's CPE Reference Implementation of the procedures specified in NIST IR-7695 for binding and unbinding WFNs.

Matching

- The CPE 2.3 Name Matching Specification defines a method for conducting a one-to-one comparison of a source CPE name to a target CPE name. By logically comparing CPE names as sets of values, CPE Name Matching methods can determine if common set relations hold. For example, CPE Name Matching can determine if the source and target names are equal, if one of the names is a subset of the other, or if the names are disjoint.
- One example of the value of CPE Name Matching is in determining if a particular product is installed on a system. Suppose that an organization is identifying which of its systems have any variation of Microsoft Internet Explorer 8 installed. This could be represented with the following well-formed CPE name (WFN):
 - `wfn:[part="a",vendor="microsoft",product="internet_explorer",version="8\.*",update=ANY,edition=ANY,language=ANY]`
- An asset management tool could collect information on the software installed on a system and compare its Internet Explorer installation characteristics to the WFN above. Suppose that the WFN for a particular installed instance of Internet Explorer was reported as:
 - `wfn:[part="a",vendor="microsoft",product="internet_explorer",version="8\.0\.6001",update=NA,edition=NA,language="en\-us"]`

Matching

- Using these two example WFNs, CPE Name Matching methods perform a pairwise comparison of attribute values in the first (source) WFN to those in the second (target) WFN, yielding a list of the set relations between each pair of attributes (e.g., equal, superset). This list of comparison results is then assessed, leading to a determination that the first WFN represents a superset of the second WFN. This can be interpreted to mean that the system being examined does indeed have a variation of Microsoft Internet Explorer 8 installed.
- Although this may seem like a trivial example, CPE Name Matching is a powerful and flexible way of performing product comparisons in a standardized, automated manner. CPE Name Matching is also used by other CPE specifications to conduct more complex tasks, such as searching for product names in CPE dictionaries and performing complex comparisons of sets of product versions—for example, determining if a system is running a particular operating system version, running two particular applications, and not running a third particular application.
- **CPE 2.3 Name Matching Specification**
- Go to the [Downloads](#) section below to download the entire CPE 2.3 Naming Matching Specification document, NIST IR-7696.

Dictionary

- The CPE 2.3 Dictionary Specification defines a standardized method for creating and managing CPE dictionaries. A *dictionary* is a repository of CPE names and metadata associated with the names. Each CPE name in the dictionary identifies a single class of IT product in the world. The word "class" here signifies that the object identified is not a physical instantiation of a product on a system, but rather the abstract model of that product. Although organizations may use a CPE name to represent either a single product class or a set of multiple product classes, a CPE dictionary stores only bound forms of well-formed CPE names (WFNs) that identify a single product class, not a set of product classes. These single product-class WFNs in bound form are referred to as *identifier names*. An example of a WFN and its bound forms is shown below.
 - WFN:wfn:[part="o",vendor="microsoft",product="windows_vista",version="6\o",update="sp1",edition=NA,language=NA,sw_edition="home_premium",target_sw=NA,target_hw="x64",other=NA] WFN bound to a URL:cpe:/o:microsoft:windows_vista:6.o:sp1:---home_premium---x64~- WFN bound to a formatted string:cpe:2.3:o:microsoft:windows_vista:6.o:sp1:-::-home_premium:-:x64:-
Official CPE Dictionary
- NIST hosts the [Official CPE Dictionary](#), which is the authoritative repository of identifier names. The authoritative nature of the Official CPE Dictionary allows organizations to search for and find identifier names in one centralized place without worrying about dealing with conflicts between federated dictionaries.

Dictionary

- Dictionary allows organizations to search for and find identifier names in one centralized place without worrying about dealing with conflicts between federated dictionaries.
- **Other CPE Dictionaries**
- Organizations may create their own extended CPE dictionaries, which are used to store identifier names not present in the Official CPE Dictionary. There are several reasons why extended dictionaries are needed. For example, an organization may have to create identifier names for proprietary products that are only useful within that organization, such as internally developed software not found outside the organization. Another possible reason is that an IT company may want to use identifiers for their unreleased products that do not yet have official identifier names; once the identifier names have stabilized, the company would submit them to the Official CPE Dictionary.
- **Adding New CPE Identifiers to the Dictionary**
- An organization may discover new products in use that do not yet have official identifiers; the organization could create identifiers, add them to its own extended dictionary, submit them for inclusion in the [Official CPE Dictionary](#), and use them internally while waiting for their addition to the Official CPE Dictionary.
- See the [CPE Dictionary page](#) on this website for an overview of the submission process.
- **CPE 2.3 Dictionary Specification**
- Go to the [Downloads](#) section below to download the entire CPE 2.3 Dictionary Specification document, NIST IR-7698.

Applicability Language

- The CPE 2.3 Applicability Language Specification defines a standardized way to describe IT platforms by forming complex logical expressions out of individual CPE names and references to checks. For example, one could use the CPE 2.3 Applicability Language to combine the CPE name for an operating system (such as Microsoft Windows XP), the CPE name for an application running on that operating system (such as Microsoft Office 2007), and a reference to a check for a particular value of a certain configuration setting (such as the wireless network card being enabled in the operating system). These logical expressions are called *applicability statements*, because they are used to designate which platforms particular guidance, policies, etc. apply to. Applicability statements can be used by tools to determine whether a target system is an instance of a particular platform.

Applicability Language

- The CPE names used by the CPE 2.3 Applicability Language specification are bound forms of well-formed CPE names (WFNs), which are the abstract logical constructions for CPE names. The basic building block of the CPE 2.3 Applicability Language Specification is referred to as the logical test. This is a logical conjunction (AND) or disjunction (OR) of one or more CPE names and/or references to checks. Individual logical tests can also be negated (inverted). Nested logical tests allow the user to express a platform as any logical combination of individual CPE names and/or references to checks.
- Note that previous versions of CPE referred to the Applicability Language Specification as simply the Language specification.
- **CPE 2.3 Applicability Language Specification**
- Go to the [Downloads](#) section below to download the entire CPE 2.3 Applicability Language Specification document, NIST IR-7697.

Applicability Language

- The CPE 2.3 Applicability Language Specification defines a standardized way to describe IT platforms by forming complex logical expressions out of individual CPE names and references to checks. For example, one could use the CPE 2.3 Applicability Language to combine the CPE name for an operating system (such as Microsoft Windows XP), the CPE name for an application running on that operating system (such as Microsoft Office 2007), and a reference to a check for a particular value of a certain configuration setting (such as the wireless network card being enabled in the operating system). These logical expressions are called *applicability statements*, because they are used to designate which platforms particular guidance, policies, etc. apply to. Applicability statements can be used by tools to determine whether a target system is an instance of a particular platform.

Applicability Language

- The CPE names used by the CPE 2.3 Applicability Language specification are bound forms of well-formed CPE names (WFNs), which are the abstract logical constructions for CPE names. The basic building block of the CPE 2.3 Applicability Language Specification is referred to as the logical test. This is a logical conjunction (AND) or disjunction (OR) of one or more CPE names and/or references to checks. Individual logical tests can also be negated (inverted). Nested logical tests allow the user to express a platform as any logical combination of individual CPE names and/or references to checks.

Specifications — Version 2.3

The following CPE 2.3 Specifications are available to view or download from NIST's Computer Security Resource Center:

- [NIST IR 7695 — Common Platform Enumeration: Naming Specification Version 2.3](#) (PDF, 1,047 KB)
- [NIST IR 7696 — Common Platform Enumeration: Name Matching Specification Version 2.3](#) (PDF, 1,100 KB)
- [NIST IR 7697 — Common Platform Enumeration: Dictionary Specification Version 2.3](#) (PDF, 896 KB)
- [NIST IR 7698 — Common Platform Enumeration: Applicability Language Specification Version 2.3](#) (PDF, 898 KB)

Reference Implementation — Version 2.3

MITRE has created a reference implementation of the CPE Naming and Matching algorithms, as described in NIST IRs [7695](#) and [7696](#). Note that the ZIP file below contains a readme.txt file that explains how to build and run the application.

- [CPE Reference Implementation](#) (ZIP, 97 KB)

CPE Dictionary

The CPE Dictionary is the official collection of CPE Names. Its purposes are to:

- Provide a canonical source for all known CPE Names.
- Bind descriptive metadata (such as a title and notes) to a CPE Name.
- Bind diagnostic tests (such as an automated check to determine if a given platform matches the name) to a CPE Name.
- The [CPE Dictionary](#) is hosted and maintained by the [National Institute for Standards and Technology \(NIST\)](#) as part of the [U.S. National Vulnerability Database \(NVD\)](#) program. NIST is responsible for ensuring that the CPE Dictionary conforms to the [CPE Specifications](#) (currently at version 2.3), and for managing the content review and quality assurance processes.

CPE Dictionary

Submission Process

- CPE Names (also called CPE Identifiers, CPE-IDs, and CPEs) are created by the [CPE Community](#) on an as-needed basis, and must be well-formed as determined by the appropriate [CPE Specifications](#). Community members who require a new CPE Name should create a CPE-conformant name and submit it to the CPE Team for inclusion in the CPE Dictionary. The process is as follows:
- Refer to the appropriate CPE Specifications to develop your proposed CPE Name(s), making your best effort to create well-formed names. It is also recommended that you refer to the current [CPE Dictionary](#) for examples and precedents.
- Prepare each CPE Name submission in valid CPE Dictionary XML form as defined by the [CPE 2.2 Dictionary Schema](#), which is also included by NIST on its CPE Dictionary page. You may bundle multiple name submissions into a single XML file. We recommend that you run an XML Schema validator to ensure your submission validates against the CPE Dictionary Schema.
- Send your submission via email to cpe_dictionary@nist.gov.
- NIST will conduct a quality assurance review before adding new names to the CPE Dictionary. During this review, portions of a proposed CPE Name(s) may be changed based on research performed against the most credible vendor and product information resources available. No name should be considered accepted until it is published in the CPE Dictionary.
- NIST will notify you when new CPE Names have been reviewed and approved.
- Error reports and proposed content updates to the CPE Dictionary should follow the same submission process and are subject to the same QA process as new CPE Name submissions. They should be submitted in valid CPE Dictionary XML form and in a separate file from new CPE Name submissions.

PRODUCTS

CPE

Search Common Platform Enumerations (CPE)

This search engine can perform a keyword search, or a CPE Name search. The keyword search will perform searching across all components of the CPE name for the user specified search text. The CPE Name search will perform searching for an exact match, as well as searching for all records that contain the components specified in the user-specified CPE Name.

CPE Naming Format: 2.3 2.2

CPE Name or Keyword:

Submit

Include deprecated CPEs

PRODUCTS

CPE

SEARCH

Q Search Results (Refine Search)

Search Parameters:

- Keyword: Internet Explorer
- CPE Status: FINAL
- CPE Naming Format: 2.3

There are **202** matching records.

Displaying matches **1** through **20**.

1 2 3 4 5 6 7 8 9 10 > >>

Vendor	Product	Version	Update	Edition	Language
cpe:2.3:a:adobe:flash_player:18.0:*:*:*:*:internet_explorer_10:*:* <small>View CVEs</small>	adobe	flash_player	18.0		
cpe:2.3:a:adobe:flash_player:18.0:*:*:*:*:internet_explorer_11:*:* <small>View CVEs</small>	adobe	flash_player	18.0		
cpe:2.3:a:adobe:flash_player:18.0.0.203:*:*:*:*:internet_explorer_10:*:* <small>View CVEs</small>	adobe	flash_player	18.0.0.203		
cpe:2.3:a:adobe:flash_player:18.0.0.203:*:*:*:*:internet_explorer_11:*:* <small>View CVEs</small>					مشخصات سرمایه سایبری

PRODUCTS

CPE

CPE Summary

[Return to Search Listing](#)

CPE Names

Version 2.3: **cpe:2.3:a:adobe:flash_player:18.0:*:*:*:internet_explorer_10:****

Version 2.2: **cpe:/a:adobe:flash_player:18.0::~~internet_explorer_10~~**

[Read information about CPE Name encoding](#)

QUICK INFO

Created On: 08/26/2019

Last Modified On: 08/26/2019

CPE NAME COMPONENTS SELECT A COMPONENT TO SEARCH FOR SIMILAR CPES

Part: a

Vendor: adobe

Product: flash_player

Version: 18.0

Update:

Language:

Software Edition:

Target Software: internet_explorer_10

Target Hardware:

Other:

مشخصات سرمایه سایبری

8/9/2022

Part: a	Language:
Vendor: adobe	Software Edition:
Product: flash_player	Target Software: internet_explorer_10
Version: 18.0	Target Hardware:
Update:	Other:
Edition:	

Metadata

Titles:	Text	Locale
	Adobe Flash Player 18.0 for Internet Explorer 10	en_US

References:	Type	Description	URL
	Advisory		https://helpx.adobe.com/security/products/flash-player/apsb15-18.html

CPE Usage

[View Vulnerabilities](#)

NATIONAL VULNERABILITY DATABASE

VULNERABILITIES

SEARCH AND STATISTICS

Q Search Results (Refine Search)

Sort results by: Publish Date Descending Sort

Search Parameters:




There are **503** matching records.Displaying matches **1** through **20**.

- Keyword (text search):

cpe:2.3:a:adobe:flash_player:18.0:*:*:*:*:internet_explorer_10:*:*

- CPE Name Search: true

1 2 3 4 5 6 7 8 9 10 > >>

Vuln ID 	Summary 	CVSS Severity 
CVE-2020-9746	<p>Adobe Flash Player version 32.0.0.433 (and earlier) are affected by an exploitable NULL pointer dereference vulnerability that could result in a crash and arbitrary code execution. Exploitation of this issue requires an attacker to insert malicious strings in an HTTP response that is by default delivered over TLS/SSL.</p> <p>Published: October 14, 2020; 10:15:17 AM -0400</p>	<p>V3.1: 8.8 HIGH</p> <p>V2.0: 9.3 HIGH</p>
CVE-2020-3757	<p>Adobe Flash Player versions 32.0.0.321 and earlier, 32.0.0.314 and earlier, 32.0.0.321 and earlier, and 32.0.0.255 and earlier have a type confusion vulnerability. Successful exploitation could lead to arbitrary code execution.</p> <p>Published: February 13, 2020; 11:15:13 AM -0500</p>	<p>V3.1: 8.8 HIGH</p> <p>V2.0: 9.3 HIGH</p>

CVE-2018-12825	Adobe Flash Player 30.0.0.134 and earlier have a security bypass vulnerability. Successful exploitation could lead to security mitigation bypass.	V3.0: 9.8 CRITICAL V2.0: 7.5 HIGH
	Published: August 29, 2018; 9:29:01 AM -0400	
CVE-2018-4944	Adobe Flash Player versions 29.0.0.140 and earlier have an exploitable type confusion vulnerability. Successful exploitation could lead to arbitrary code execution in the context of the current user.	V3.0: 9.8 CRITICAL V2.0: 10.0 HIGH
	Published: May 19, 2018; 1:29:01 PM -0400	
CVE-2018-4933	Adobe Flash Player versions 29.0.0.113 and earlier have an exploitable out-of-bounds read vulnerability. Successful exploitation could lead to information disclosure.	V3.0: 6.5 MEDIUM V2.0: 4.0 MEDIUM
	Published: May 19, 2018; 1:29:01 PM -0400	
CVE-2018-4932	Adobe Flash Player versions 29.0.0.113 and earlier have an exploitable Use-After-Free vulnerability. Successful exploitation could lead to arbitrary code execution in the context of the current user.	V3.0: 8.8 HIGH V2.0: 9.0 HIGH
	Published: May 19, 2018; 1:29:01 PM -0400	
CVE-2018-4878	A use-after-free vulnerability was discovered in Adobe Flash Player before 28.0.0.161. This vulnerability occurs due to a dangling pointer in the Primetime SDK related to media player handling of listener objects. A successful attack can lead to arbitrary code execution. This was exploited in the wild in January and February 2018.	V3.1: 9.8 CRITICAL V2.0: 7.5 HIGH
	Published: February 06, 2018; 4:29:00 PM -0500	
CVE-2018-4877	A use-after-free vulnerability was discovered in Adobe Flash Player before 28.0.0.161. This vulnerability occurs due to a dangling pointer in the Primetime SDK related to media player's quality of service functionality. A successful attack can lead to arbitrary code execution.	V3.0: 9.8 CRITICAL V2.0: 10.0 HIGH
	Published: February 06, 2018; 4:29:00 PM -0500	
CVE-2018-4871	An Out-of-bounds Read issue was discovered in Adobe Flash Player before 28.0.0.137. This vulnerability occurs because of computation that reads data that is past the end of the target buffer. The use of an invalid (out-of-range) pointer offset during access of internal data structure fields causes	V3.0: 7.5 HIGH V2.0: 5.0 MEDIUM
	the vulnerability. A successful attack can lead to sensitive data exposure.	

[Information Technology Laboratory](#)

NATIONAL VULNERABILITY DATABASE



VULNERABILITIES

🚩 CVE-2018-12825 Detail

Current Description

Adobe Flash Player 30.0.0.134 and earlier have a security bypass vulnerability. Successful exploitation could lead to security mitigation bypass.

[+View Analysis Description](#)

Severity

CVSS Version 3.x

CVSS Version 2.0

CVSS 3.x Severity and Metrics:

8/9/2022
NIST: NVD

Base Score: **9.8 CRITICAL**

Vector: CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

QUICK INFO

CVE Dictionary Entry:

CVE-2018-12825

NVD Published Date:

08/29/2018

NVD Last Modified:

10/02/2019

Source:

Adobe Systems Incorporated

References to Advisories, Solutions, and Tools

By selecting these links, you will be leaving NIST webspace. We have provided these links to other web sites because they may have information that would be of interest to you. No inferences should be drawn on account of other sites being referenced, or not, from this page. There may be other web sites that are more appropriate for your purpose. NIST does not necessarily endorse the views expressed, or concur with the facts presented on these sites. Further, NIST does not endorse any commercial products that may be mentioned on these sites. Please address comments about this page to nvd@nist.gov.


Hyperlink	Resource
http://www.securityfocus.com/bid/105070	Third Party Advisory VDB Entry
http://www.securitytracker.com/id/1041448	Third Party Advisory VDB Entry
https://access.redhat.com/errata/RHSA-2018:2435	Third Party Advisory
https://helpx.adobe.com/security/products/flash-player/apsb18-25.html	Patch Vendor Advisory

Weakness Enumeration



CWE-ID	CWE Name	Source
NVD-CWE-noinfo	Insufficient Information	 NIST

Known Affected Software Configurations [Switch to CPE 2.2](#)




Configuration 1 [\(hide\)](#)


 cpe:2.3:a:adobe:flash_player:*:*:*:*:*:* Show Matching CPE(s)	Up to (including) 30.0.0.154
Running on/with	
cpe:2.3:o:apple:mac_os_x:*:*:*:*:* Show Matching CPE(s)	
cpe:2.3:o:linux:linux_kernel:*:*:*:*:* Show Matching CPE(s)	

Configuration 3 ([hide](#))

 cpe:2.3:a:adobe:flash_player:*:*:*:*:edge:*:* Show Matching CPE(s) ▼	Up to (including) 30.0.0.154
 cpe:2.3:a:adobe:flash_player:*:*:*:*:internet_explorer_11:*:* Show Matching CPE(s) ▼	Up to (including) 30.0.0.154
Running on/with	
cpe:2.3:o:microsoft:windows_10:-:*:*:*:*:* Show Matching CPE(s) ▼	
cpe:2.3:o:microsoft:windows_8.1:-:*:*:*:*:* Show Matching CPE(s) ▼	

Configuration 4 ([hide](#))

 cpe:2.3:o:redhat:enterprise_linux_desktop:6.0:*:*:*:*:* Show Matching CPE(s) ▼
 cpe:2.3:o:redhat:enterprise_linux_server:6.0:*:*:*:*:* Show Matching CPE(s) ▼
 cpe:2.3:o:redhat:enterprise_linux_workstation:6.0:*:*:*:*:* Show Matching CPE(s) ▼

 Denotes Vulnerable Software

Are we missing a CPE here? Please let us know.

Change History

3 change records found [show changes](#)

8/9/2022

مشخصات سرمایه سایبری

Search Results (Refine Search)

Search Parameters:

- Keyword: window 2018
- CPE Status: FINAL
- CPE Naming Format: 2.3

There are **81** matching records.

Displaying matches **1** through **20**.

1 2 3 4 5 > >>

Vendor	Product	Version	Update	Edition	Language
cpe:2.3:a:acronis:true_image:2018:*:*:*:*:windows:*:* <small>View CVEs</small>	acronis	true_image	2018		
cpe:2.3:a:apple:icloud:7.7:*:*:*:*:windows:*:* <small>View CVEs</small>	apple	icloud	7.7		
cpe:2.3:a:apple:icloud:7.8.1:*:*:*:*:windows:*:* <small>View CVEs</small>	apple	icloud	7.8.1		
cpe:2.3:a:atlassian:sourcetree:0.5a:*:*:*:*:windows:*:* <small>View CVEs</small>	atlassian	sourcetree	0.5a		
cpe:2.3:a:beyondtrust:avecto_defendpoint:4.0.191.0:*:*:*:*:* <small>View CVEs</small>	beyondtrust	avecto_defendpoint	4.0.191.0		
cpe:2.3:a:beyondtrust:avecto_defendpoint:4.0.247:*:*:*:*:* <small>View CVEs</small>	beyondtrust	avecto_defendpoint	4.0.247		

VULNERABILITIES

SEARCH AND STATISTICS

Search Results (Refine Search)

Sort results by: Publish Date Descending Sort

Search Parameters:

There are **1** matching records.
 Displaying matches **1** through **1**.



- Results Type: Overview
- Keyword (text search):
cpe:2.3:a:acronis:true_image:2018:*:*:*:windows:*
- CPE Name Search: true

Vuln ID 	Summary 	CVSS Severity 
CVE-2020-35145	Acronis True Image for Windows prior to 2021 Update 3 allowed local privilege escalation due to a DLL hijacking vulnerability in multiple components, aka an Untrusted Search Path issue. Published: January 29, 2021; 2:15:17 AM -0500	V3.1: 7.8 HIGH V2.0: 4.4 MEDIUM

CPE Dictionary

- [Official CPE Dictionary v2.3, gz format](#) - 14.36 MB, Updated: 08/02/2022; 12:36:47 AM -0400
- [Official CPE Dictionary v2.3, zip format](#) - 14.36 MB, Updated: 08/02/2022; 12:36:47 AM -0400
- [Official CPE Dictionary v2.2, gz format](#) - 17.74 MB, Updated: 08/02/2022; 12:36:47 AM -0400
- [Official CPE Dictionary v2.2, zip format](#) - 17.74 MB, Updated: 08/02/2022; 12:36:47 AM -0400
- [CPE Dictionary Search](#)
- [CPE Dictionary Growth Statistics](#)
- **CPE Standards Information**
- General information on [CPE](#)
- The [CPE 2.3 XML Schema](#)
- The [CPE 2.3 Dictionary Extension XML Schema](#)
- The [CPE 2.2 XML Schema](#)
- **NIST Dictionary CPE Repository Metadata**
- The [NIST CPE Metadata 0.2 XML Schema](#)

[Show Matching CPE\(s\)](#)▼**Configuration 3** ([hide](#))

 cpe:2.3:a:adobe:flash_player:*:*:*:*:edge:*:* Show Matching CPE(s) ▼	Up to (including) 30.0.0.154
 cpe:2.3:a:adobe:flash_player:*:*:*:*:internet_explorer_11:*:* Show Matching CPE(s) ▼	Up to (including) 30.0.0.154


Running on/with


cpe:2.3:o:microsoft:windows_10:-:*:*:*:*:*
[Show Matching CPE\(s\)](#)▼

cpe:2.3:o:microsoft:windows_8.1:-:*:*:*:*:*
[Show Matching CPE\(s\)](#)▼

Configuration 4 ([hide](#))

 **cpe:2.3:o:redhat:enterprise_linux_desktop:6.0:*:*:*:*:***
[Show Matching CPE\(s\)](#)▼

 **cpe:2.3:o:redhat:enterprise_linux_server:6.0:*:*:*:*:***
[Show Matching CPE\(s\)](#)▼

 **cpe:2.3:o:redhat:enterprise_linux_workstation:6.0:*:*:*:*:***
[Show Matching CPE\(s\)](#)▼

 Denotes Vulnerable Software

Are we missing a CPE here? Please let us know.

Change History

8/9/2022
3 change records found [show changes](#)

Common Platform Enumeration (CPE) Dictionary Statistics

- CPE is a structured naming scheme for information technology systems, software, and packages.
- Based upon the generic syntax for Uniform Resource Identifiers (URI), CPE includes a formal name format, a method for checking names against a system, and a description format for binding text and tests to a name.

text and tests to a name.

The below statistics capture growth and modification information relating to the official CPE Product Dictionary.

**The CPE Dictionary Team is working to address any discrepancy between the number of processed and received submissions. Some submissions may be delayed because they require a feedback cycle between the CPE Dictionary Team and CPE Dictionary Submitters to ensure consistency within the CPE dictionary.*

2022

Month	New Entries	Modified Entries	New Vendors	New Products	Deprecated Entries
January	11,326	11,767	201	1,104	1,006
February	12,182	12,312	112	1,626	1,076
March	13,651	14,716	196	1,786	1,009
April	10,468	14,202	189	1,483	2,083
May	10,322	10,581	240	1,747	370
June	14,508	15,124	300	1,362	1,142
July	13,945	17,088	282	1,136	1,167
August	601	799	12	40	7

2021

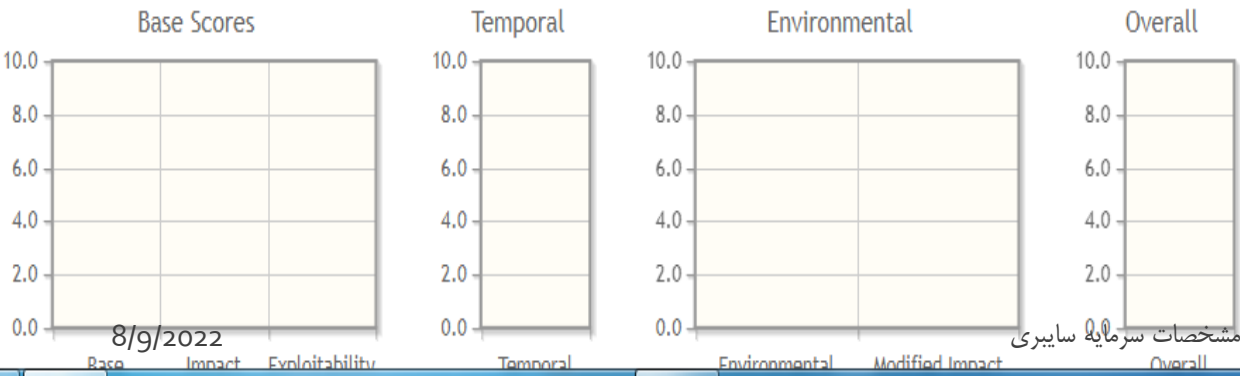
Month	New Entries	Modified Entries	New Vendors	New Products	Deprecated Entries	
January	11,253	10,836	186	1,716	1,132	
February	14,969	11,590	220	1,882	823	
March	18,562	20,634	113	1,282	803	
April	15,477	35,836	167	721	1,715	
May	14,001	31,245	135	1,123	1,325	
June	18,734	27,364	143	1,808	909	
July	20,798	23,361	199	898	1,838	
August	8/9/2022	16,553	20,568	مشخصات نه‌ایه سایبری	830	1,738

VULNERABILITY METRICS

CVSS Version 3.0 **CVSS Version 3.1**

Common Vulnerability Scoring System Calculator

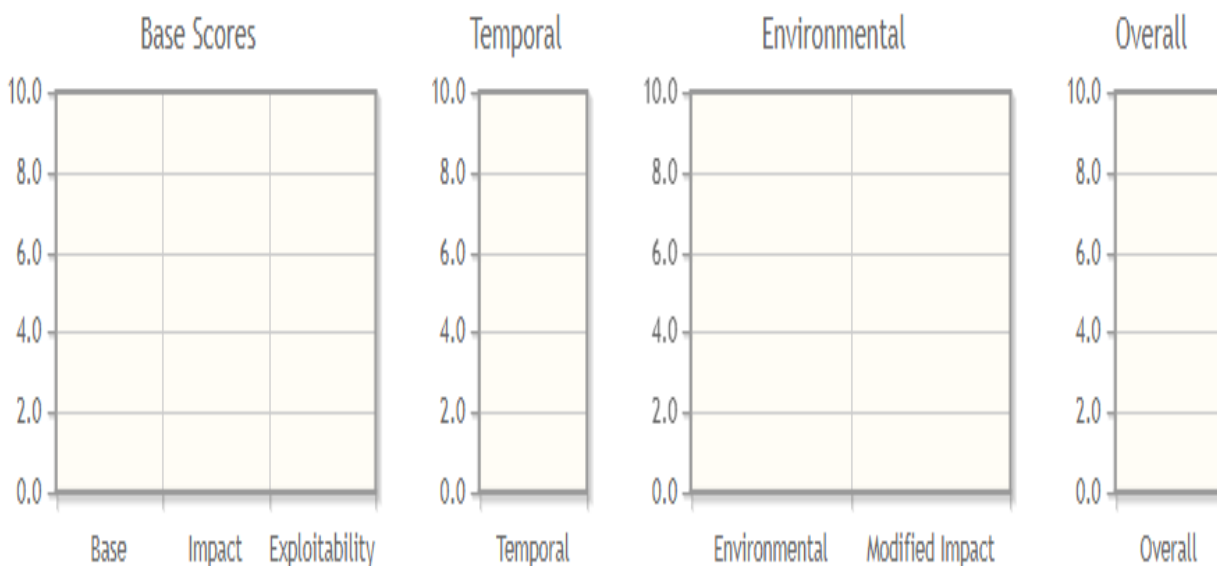
This page shows the components of the CVSS score for example and allows you to refine the CVSS base score. Please read the CVSS standards guide to fully understand how to score CVSS vulnerabilities and to interpret CVSS scores. The scores are computed in sequence such that the Base Score is used to calculate the Temporal Score and the Temporal Score is used to calculate the Environmental Score.



CVSS Base Score: NA
 Impact Subscore: NA
 Exploitability Subscore: NA
CVSS Temporal Score: NA
 CVSS Environmental Score: NA
 Modified Impact Subscore: NA
Overall CVSS Score: NA

Common Vulnerability Scoring System Calculator

This page shows the components of the CVSS score for example and allows you to refine the CVSS base score. Please read the CVSS standards guide to fully understand how to score CVSS vulnerabilities and to interpret CVSS scores. The scores are computed in sequence such that the Base Score is used to calculate the Temporal Score and the Temporal Score is used to calculate the Environmental Score.



CVSS Base Score: NA
Impact Subscore: NA
Exploitability Subscore: NA
CVSS Temporal Score: NA
CVSS Environmental Score: NA
Modified Impact Subscore: NA
Overall CVSS Score: NA

Show Equations

CVSS v3.1 Vector

NA مشخصات سرمایه سایبری

CVSS

Base Score Metrics

Exploitability Metrics

Attack Vector (AV)*

Network (AV:N) Adjacent Network (AV:A) Local (AV:L) Physical (AV:P)

Attack Complexity (AC)*

Low (AC:L) High (AC:H)

Privileges Required (PR)*

None (PR:N) Low (PR:L) High (PR:H)

User Interaction (UI)*

None (UI:N) Required (UI:R)

Scope (S)*

Unchanged (S:U) Changed (S:C)

Impact Metrics

Confidentiality Impact (C)*

None (C:N) Low (C:L) High (C:H)

Integrity Impact (I)*

None (I:N) Low (I:L) High (I:H)

Availability Impact (A)*

None (A:N) Low (A:L) High (A:H)

CVSS v3.1 Equations

- The CVSS v3.1 equations are defined below.
- **Base**
- The Base Score is a function of the Impact and Exploitability sub score equations. Where the Base score is defined as,
 - $$\begin{array}{ll} \text{If (Impact sub score} \leq 0) & 0 \text{ else,} \\ \text{Scope Unchanged}_4 & \text{Roundup}(\text{Minimum}[(\text{Impact} + \text{Exploitability}), 10]) \\ \text{Scope Changed} & \text{Roundup}(\text{Minimum}[1.08 \times (\text{Impact} + \text{Exploitability}), 10]) \end{array}$$

and the Impact sub score (ISC) is defined as,

$$\begin{array}{ll} \text{Scope Unchanged} & 6.42 \times ISC_{\text{Base}} \\ \text{Scope Changed} & 7.52 \times [ISC_{\text{Base}} - 0.029] - 3.25 \times [ISC_{\text{Base}} - 0.02]^{15} \end{array}$$

Where,

$$ISC_{\text{Base}} = 1 - [(1 - \text{Impact}_{\text{Conf}}) \times (1 - \text{Impact}_{\text{Integ}}) \times (1 - \text{Impact}_{\text{Avail}})]$$

And the Exploitability sub score is,

$$8.22 \times \text{AttackVector} \times \text{AttackComplexity} \times \text{PrivilegeRequired} \times \text{UserInteraction}$$

CVSS

Temporal Score Metrics

Exploit Code Maturity (E)

Not Defined (E:X)	Unproven that exploit exists (E:U)	Proof of concept code (E:P)	Functional exploit exists (E:F)	High (E:H)
-------------------	------------------------------------	-----------------------------	---------------------------------	------------

Remediation Level (RL)

Not Defined (RL:X)	Official fix (RL:O)	Temporary fix (RL:T)	Workaround (RL:W)	Unavailable (RL:U)
--------------------	---------------------	----------------------	-------------------	--------------------

Report Confidence (RC)

Not Defined (RC:X)	Unknown (RC:U)	Reasonable (RC:R)	Confirmed (RC:C)
--------------------	----------------	-------------------	------------------

CVSS v3.1 Equations

- **Temporal**
- The Temporal score is defined as,

Roundup(BaseScore × ExploitCodeMaturity × RemediationLevel × ReportConfidence)

CVSS

Environmental Score Metrics

Exploitability Metrics

Attack Vector (MAV)

Not Defined (MAV:X) Network (MAV:N) Adjacent Network (MAV:A)

Local (MAV:L) Physical (MAV:P)

Attack Complexity (MAC)

Not Defined (MAC:X) Low (MAC:L) High (MAC:H)

Privileges Required (MPR)

Not Defined (MPR:X) None (MPR:N) Low (MPR:L) High (MPR:H)

User Interaction (MUI)

Not Defined (MUI:X) None (MUI:N) Required (MUI:R)

Scope (MS)

Not Defined (MS:X) Unchanged (MS:U) Changed (MS:C)

Impact Metrics

Confidentiality Impact (MC)

Not Defined (MC:X) None (MC:N) Low (MC:L)

High (MC:H)

Integrity Impact (MI)

Not Defined (MI:X) None (MI:N) Low (MI:L)

High (MI:H)

Availability Impact (MA)

Not Defined (MA:X) None (MA:N) Low (MA:L)

High (MA:H)

Impact Subscore Modifiers

Confidentiality Requirement (CR)

Not Defined (CR:X) Low (CR:L)

Medium (CR:M) High (CR:H)

Integrity Requirement (IR)

Not Defined (IR:X) Low (IR:L) Medium (IR:M)

High (IR:H)

Availability Requirement (AR)

Not Defined (AR:X) Low (AR:L)

Medium (AR:M) High (AR:H)

CVSS v3.1 Equations

- **Environmental**

- The environmental score is defined as,

If (Modified Impact Sub score ≤ 0) 0 else,

If Modified Scope is Unchanged $\text{Round up}(\text{Round up}(\text{Minimum}[(M.\text{Impact} + M.\text{Exploitability}), 10]) \times \text{Exploit Code Maturity} \times \text{Remediation Level} \times \text{Report Confidence})$

If Modified Scope is Changed $\text{Round up}(\text{Round up}(\text{Minimum}[1.08 \times (M.\text{Impact} + M.\text{Exploitability}), 10]) \times \text{Exploit Code Maturity} \times \text{Remediation Level} \times \text{Report Confidence})$

And the modified Impact sub score is defined as,

If Modified Scope is Unchanged $6.42 \times [ISC_{\text{Modified}}]$

If Modified Scope is Changed $7.52 \times [ISC_{\text{Modified}} - 0.029] - 3.25 \times [ISC_{\text{Modified}} \times 0.9731 - 0.02]^{13}$

Where,

$ISC_{\text{Modified}} = \text{Minimum} [[1 - (1 - M. IConf \times CR) \times (1 - M. IInteg \times IR) \times (1 - M. IAvail \times AR)], 0.915]$

The Modified Exploitability sub score is,

$8.22 \times M. \text{AttackVector} \times M. \text{AttackComplexity} \times M. \text{PrivilegeRequired} \times M. \text{UserInteraction}_4$ Where "Round up" is defined as the smallest number, specified to one decimal place, that is equal to or higher than its input. For example, Round up (4.02) is 4.1; and Round up (4.00) is 4.0.

CVSS

NVD CVSS Calculators

[NVD CVSS v2 Calculator](#)

[NVD CVSS v3 Calculator](#)

NVD Vulnerability Severity Ratings

NVD provides qualitative severity ratings of "Low", "Medium", and "High" for CVSS v2.0 base score ranges in addition to the severity ratings for CVSS v3.0 as they are defined in the CVSS v3.0 specification.

CVSS v2.0 Ratings		CVSS v3.0 Ratings	
Severity	Base Score Range	Severity	Base Score Range
		None	0.0
Low	0.0-3.9	Low	0.1-3.9
Medium	4.0-6.9	Medium	4.0-6.9
High	7.0-10.0	High	7.0-8.9
		Critical	9.0-10.0

Security Content Automation Protocol (SCAP)

- The Security Content Automation Protocol (SCAP) is a synthesis of interoperable specifications derived from community ideas. Community participation is a great strength for SCAP, because the security automation community ensures the broadest possible range of use cases is reflected in SCAP functionality. This Web site is provided to support continued community involvement. From this site, you will find information about both [existing SCAP specifications](#) and [emerging specifications](#) relevant to NIST's security automation agenda. You are invited to participate, whether monitoring community dialog or leading more substantive activities like specification authorship.
- NIST's security automation agenda is broader than the vulnerability management application of modern day SCAP. Many different security activities and disciplines can benefit from standardized expression and reporting. We envision further expansion in compliance, remediation, and network monitoring, and encourage your contribution relative to these and additional disciplines. NIST is also working on this expansion plan, so please communicate with the [SCAP Team](#) early and often to ensure proper coordination of efforts.

Asset Identification

- Asset identification plays an important role in an organization's ability to quickly correlate different sets of information about assets. This specification provides the necessary constructs to uniquely identify assets based on known identifiers and/or known information about the assets. This specification describes the purpose of asset identification, a data model for identifying assets, methods for identifying assets, and guidance on how to use asset identification. It also identifies a number of known use cases for asset identification.
- The [Asset Specifications Development List](#) is available for developers interested in Asset Identification and other asset related security automation standards. Please subscribe to this list through the [SCAP Community](#) page.

Asset Identification

- **Asset Identification Resources**
- [Release 1.1](#)
- [Release 1.0 \(Early Access 1\)](#)

Asset Identification 1.1 Resources (June 15, 2011)

XML Schema Files: [[what is a schema?](#)]

[Asset Identification 1.1 Schema](#) (XSD 1.0)

Documentation:

[NISTIR 7693](#)

Asset Reporting Format (ARF)

- The Asset Reporting Format (ARF) is a data model to express the transport format of information about assets, and the relationships between assets and reports. The standardized data model facilitates the reporting, correlating, and fusing of asset information throughout and between organizations. ARF is vendor and technology neutral, flexible, and suited for a wide variety of reporting applications.
- The [Emerging Specifications Discussion List](#) is available for developers interested in ARF and other emerging security automation standards. Please subscribe to this list through the [SCAP Community](#) page.
- ARF Resources
- [Release 1.1](#)
- [Release 1.0 \(Early Access 1\)](#)
- ARF 1.1 Resources (June 21, 2011)
- XML Schema Files: [[what is a schema?](#)]
- [ARF 1.1 Schema](#) (XSD 1.0)
- Documentation:
- [NISTIR 7694](#)

Common Configuration Enumeration (CCE)

- The [CCE List](#) provides unique identifiers to security-related system configuration issues in order to improve workflow by facilitating fast and accurate correlation of configuration data across multiple information sources and tools.
- For example, CCE Identifiers are included for the settings in [Microsoft Corporation's Windows Server 2008 Security Guide](#) and [2007 Microsoft Office Security Guide](#); are the main identifiers used for the settings in the U.S. [Federal Desktop Core Configuration \(FDCC\)](#) data file downloads; and provide a mapping between the elements in configuration best-practice documents including the [Center for Internet Security's \(CIS\) CIS Benchmark Documents](#), [National Institute of Standards and Technology's \(NIST\) NIST Security Configuration Guides](#), [National Security Agency's \(NSA\) NSA Security Configuration Guides](#), and [Defense Information Systems Agency's \(DISA\) DISA Security Technical Implementation Guides \(STIGS\)](#).
- When dealing with information from multiple sources, use of consistent identifiers can improve data correlation; enable interoperability; foster automation; and ease the gathering of metrics for use in situation awareness, IT security audits, and regulatory compliance. For example, [Common Vulnerabilities and Exposures \(CVE®\)](#) provides this capability for information security vulnerabilities.

Common Configuration Enumeration (CCE)

- Similar to the CVE effort, CCE assigns a unique, common identifier to a particular security-related configuration issue. CCE identifiers are associated with configuration statements and configuration controls that express the way humans name and discuss their intentions when configuring computer systems. In this way, the use of CCE-IDs as tags provide a bridge between natural language, prose-based configuration guidance documents and machine-readable or executable capabilities such as configuration audit tools.
- Each entry on the [CCE List](#) contains the following five attributes:
 - CCE Identifier Number – "CCE-2715-1"
 - Description – a humanly understandable description of the configuration issue
 - Conceptual Parameters – parameters that would need to be specified in order to implement a CCE on a system
 - Associated Technical Mechanisms – for any given configuration issue there may be one or more ways to implement the desired result
 - References – pointers to the specific sections of the documents or tools in which the configuration issue is described in detail
 - Currently, CCE is focused solely on software-based configurations. Recommendations for hardware and/or physical configurations are not supported. Refer to the [CCE List](#) for more information.

NATIONAL CHECKLIST PROGRAM

General

Checklist Repository

Data Feeds

Data Mappings

Participant Material

CCE

Contact NCP

Other Sites

+ CCE Platform Listing

Common Configuration Enumeration (CCE) provides unique identifiers to system configuration issues in order to facilitate fast and accurate correlation of configuration data across multiple information sources and tools. For example, CCE Identifiers can be used to associate checks in configuration assessment tools with statements in configuration best-practice.

+ CCE Submissions, comments and questions can be sent to cce@nist.gov.

+ CCE List

The current release of CCE is 5.20220713 (CCE Version 5, updated on July 13, 2022). A ChangeLog is available that details the changes since the last release, 5.20210407.

+ DOWNLOADS (MS Excel format) DATE UPDATED

CCE v5 - All Platform Groups-COMBINED FILE (Excel) (1999 KB)	July 13, 2022
CCE v5 - Apple macOS Bigsur (347 KB)	July 13, 2022
CCE v5 - Apple macOS Catalina (297 KB)	July 13, 2022
CCE v5 - Apple macOS Monterey (353 KB)	July 13, 2022
CCE v5 - SUSE Linux Enterprise Server 15 (131 KB)	July 13, 2022

Open Vulnerability and Assessment Language

- **OVAL**® International in scope and free for public use, OVAL is an information security community effort to standardize how to assess and report upon the machine state of computer systems. OVAL includes a language to encode system details, and an assortment of content repositories held throughout the community.
- Tools and services that use OVAL for the three steps of system assessment — representing system information, expressing specific machine states, and reporting the results of an assessment — provide enterprises with accurate, consistent, and actionable information so they may improve their security. Use of OVAL also provides for reliable and reproducible information assurance metrics and enables interoperability and automation among security tools and services.

Open Vulnerability and Assessment Language

- [Vulnerability Assessment](#)
- [Configuration Management](#)
- [Patch Management](#)
- [Policy Compliance](#)
- [Community Repositories of OVAL Content](#)
- [Vulnerability Databases and Advisories](#)
- [Benchmark Writing](#)
- [Security Content Automation](#)

Open Checklist Interactive Language (OCIL)

- The Open Checklist Interactive Language (OCIL) defines a framework for expressing a set of questions to be presented to a user and corresponding procedures to interpret responses to these questions. Although the OCIL specification was developed for use with IT security checklists, the uses of OCIL are by no means confined to IT security. Other possible use cases include research surveys, academic course exams, and instructional walkthroughs.
- In IT security, organizations work with security policies that detail the information that needs to be secured and the security requirements that must be met to ensure the information is protected accordingly. To verify compliance with security requirements, Federal agencies have already implemented security technologies that support the Security Content Automation Protocol (SCAP). OCIL is considered an emerging specification, so it is not currently included in SCAP. However, OCIL can still be used in conjunction with SCAP specifications such as [XCCDF](#) to help handle cases where lower-level checking languages such as OVAL are unable to automate a particular check. In short, OCIL provides a standardized approach to express and evaluate non-automated (i.e., manual) security checks.

Open Checklist Interactive Language (OCIL)

- OCIL provides the conceptual framework for representing non-automatable questions. The following list defines the features supported by OCIL:
- Ability to define questions (of type Boolean, Choice, Numeric, or String)
- Ability to define possible answers to a question from which the user can choose
- Ability to define actions to be taken resulting from a user's answer
- Ability to enumerate the result set

Trust Model for Security Automation Data (TMSAD)

- TMSAD describes a common trust model that can be applied to specifications within the security automation domain, such as Security Content Automation Protocol (SCAP). Since information in the security automation domain is primarily exchanged using Extensible Markup Language (XML), the focus of this model is on the processing of XML documents. The trust model is composed of recommendations on how to use existing specifications to represent signatures, hashes, key information, and identity information in the context of an XML document within the security automation domain.

Extensible Configuration Checklist Description Format (XCCDF)

- XCCDF 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.
- XCCDF documents are expressed in XML, and may be validated with an XML Schema-validating parser.

Software Identification (SWID)

Tagging SWID

- Software is vital to our economy and way of life as part of the critical infrastructure for the modern world. Too often cost and complexity make it difficult to manage software effectively, leaving the software open for attack. To properly manage software, enterprises need to maintain accurate software inventories of their managed devices in support of higher-level business, information technology, and cybersecurity functions. Accurate software inventories help an enterprise to:
 - **Manage compliance with software license agreements.** Knowing what software is installed and used can help an enterprise to avoid paying for unneeded licenses.
 - **Ensure that all software assets in use conform to organizational policy.** Reducing and controlling an organization's software footprint can reduce the surface area of attack.

Software Identification (SWID)

Tagging SWID

- **Verify that all deployed software assets are updated and free of known exploitable weaknesses.** Ensuring all software is patched and updated is an effective way to counter cyber threats.
- **Support assessing that all deployed software assets are configured according to their organizations' security policies.** Configuring defensive mechanisms, reducing services exposed, and restricting features used within software can also further reduce attack surface, and can harden systems against attacks. Accurate software inventories identify critical software assets so that assessments can be targeted and tracked.
- **Plan software investments and resources needed to support upgrades to and replacement of legacy systems.** Knowledge of what commercial and custom software the enterprise uses can assist with budgeting for IT investments.

Software Identification (SWID)

Tagging SWID

- While some vendors provide tools to manage licenses, updates, patches, and configurations for their products, organizations need to monitor and use many such tools to address the range of products their enterprises deploy. This multiplicity of tools creates an environment where human error and a lack of resources can limit an enterprise's ability to support the active management of software, preventing timely patching, and allowing configurations to drift and software licenses to be inefficiently utilized. Instead, a single mechanism is needed that can help organizations to understand the state of all software across their enterprise regardless of vendor.
- Software Identification (SWID) Tags, defined by the ISO/IEC 19770-2:2015 standard, promise to be an important step towards such a goal. SWID Tags provide a transparent way for organizations to track the software installed on their managed devices. SWID Tag files contain descriptive information about a specific release of a software product. The SWID standard defines a [lifecycle](#) where a SWID Tag is added to an endpoint as part of the software product's installation process and deleted by the product's uninstall process. When this lifecycle is followed, the presence of a given SWID Tag corresponds directly to the presence of the software product that the Tag describes. The National Institute of Standards and Technology recommends adoption of the SWID Tag standard by software producers, and multiple standards bodies, including the Trusted Computing Group (TCG) and the Internet Engineering Task Force (IETF) utilize SWID Tags in their standards.

Software Identification (SWID)

Tagging SWID

- NIST plans to continue to promote the incorporation of the SWID Tag standard and [associated guidelines](#) in other international consensus standards (such as IETF and TCG efforts), the broad adoption of SWID tagging within the software community, and the use of SWID Tag information in the creation of cybersecurity reference data and [security automation](#) content. Additionally, NIST is working to incorporate SWID Tag data into the vulnerability dataset provided by [National Vulnerability Database](#) (NVD), and has incorporated use of SWID Tag data into [Security Content Automation Protocol](#) (SCAP) version 1.3. These efforts are part of the Software Identification (SWID) Tagging project, which is an initiative of the Computer Security Division's Security Automation Program (SAP). The SAP is focused on standardizing the exchange of security posture information supporting the management of software, vulnerabilities, patches, and secure configurations for computing devices.

با تشکر از توجه شما

اللهم صل على محمد وآل محمد