NISTIR 8085 (DRAFT)

Forming Common Platform Enumeration (CPE) Names from Software Identification (SWID) Tags

David Waltermire
Brant A. Cheikes

National Institute of
Standards and Technology
U.S. Department of Commerce

NISTIR 8085 (DRAFT)

Forming Common Platform Enumeration (CPE) Names from Software Identification (SWID) Tags

David Waltermire
Computer Security Division
Information Technology Laboratory

Brant A. Cheikes
The MITRE Corporation
Bedford, Massachusetts

December 2015

U.S. Department of Commerce
Penny Pritzker, Secretary

National Institute of Standards and Technology
Willie May, Under Secretary of Commerce for Standards and Technology and Director

National Institute of Standards and Technology Internal Report 8085
18 pages (December 2015)

Certain commercial entities, equipment, or materials may be identified in this document in order to describe an experimental procedure or concept adequately. Such identification is not intended to imply recommendation or endorsement by NIST, nor is it intended to imply that the entities, materials, or equipment are necessarily the best available for the purpose.

There may be references in this publication to other publications currently under development by NIST in accordance with its assigned statutory responsibilities. The information in this publication, including concepts and methodologies, may be used by Federal agencies even before the completion of such companion publications. Thus, until each publication is completed, current requirements, guidelines, and procedures, where they exist, remain operative. For planning and transition purposes, Federal agencies may wish to closely follow the development of these new publications by NIST.

Organizations are encouraged to review all draft publications during public comment periods and provide feedback to NIST. All NIST Computer Security Division publications, other than the ones noted above, are available at http://csrc.nist.gov/publications.

Public comment period: December 17, 2015 through January 8, 2015

National Institute of Standards and Technology
Attn: Computer Security Division, Information Technology Laboratory
100 Bureau Drive (Mail Stop 8930) Gaithersburg, MD 20899-8930
Email: nistir8060-comments@nist.gov

Note: The email used for providing public comments is the same as the email used for NISTIR 8060.

Reports on Computer Systems Technology

The Information Technology Laboratory (ITL) at the National Institute of Standards and Technology (NIST) promotes the U.S. economy and public welfare by providing technical leadership for the Nation's measurement and standards infrastructure. ITL develops tests, test methods, reference data, proof of concept implementations, and technical analyses to advance the development and productive use of information technology. ITL's responsibilities include the development of management, administrative, technical, and physical standards and guidelines for the cost-effective security and privacy of other than national security-related information in Federal information systems.

Abstract

This report describes the association between the use of Software Identification (SWID) Tags and the Common Platform Enumeration (CPE) specifications. The publication is intended as a supplement to NIST Internal Report 8060, *Guidelines for the Creation of Interoperable Software Identification (SWID) Tags*. Both SWID and CPE support automated and accurate software asset management. Such automation, in turn, helps organizations to: minimize exposure to publicly disclosed software vulnerabilities; enforce organizational policies regarding authorized software; and, control network resource access from potentially vulnerable endpoints. NISTIR 8085 provides guidance to support CPE naming using information from a SWID tag based on the International Organization for Standardization/International Electrotechnical Commission 19770-2:2015 standard.

Keywords

CPE; common platform enumeration; software; software asset management; software identification; SWID; software identification tag

Trademark Information

Any mention of commercial products or reference to commercial organizations is for information only; it does not imply recommendation or endorsement by NIST, nor does it imply that the products mentioned are necessarily the best available for the purpose.

All names are trademarks or registered trademarks of their respective owners.

Document Conventions

This report provides both informative and normative guidance supporting the use of SWID tags. The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this report are to be interpreted as described in Request for Comment (RFC) 2119. When these words appear in regular case, such as "should" or "may", they are not intended to be interpreted as RFC 2119 key words.

Some of the requirements and conventions used in this report reference Extensible Markup Language (XML) content. These references come in two forms, inline and indented. An example of an inline reference is: “One could use `<SoftwareIdentity>` `@name` as the value for the CPE “product” attribute.”

In this example, the notation `<SoftwareIdentity>` can be replaced by the more verbose equivalent “the XML element whose qualified name is SoftwareIdentity”.

The general convention used when describing XML attributes within this report is to reference the attribute as well as its associated element, employing the general form “`@attributeName` for the `<prefix:localName>`”. Indented references are intended to represent the form of actual XML content. Indented references represent literal content by the use of a fixed-length font, and parametric (freely replaceable) content by the use of an italic font. Square brackets “[]” are used to designate optional content.

Both inline and indented forms use qualified names to refer to specific XML elements. A qualified name associates a named element with a namespace. The namespace identifies the XML model, and the XML schema is a definition and implementation of that model. A qualified name declares this schema to element association using the format “*prefix*:*element-name*”. The association of prefix to namespace is defined in the metadata of an XML document and varies from document to document.

Table of Contents

List of Appendices

1 Introduction

International Organization for Standardization (ISO)/International Electrotechnical Commission (ISO/IEC) 19770-2 specifies an international standard for software identification tags, also referred to as SWID tags. A *SWID tag* is a formatted set of data elements that collectively identify and describe a software product. A significantly revised version of the standard was published in October 2015, and is designated ISO/IEC 19770-2:2015. This updated standard is referenced herein as the *SWID specification*.

NIST Internal Report 8060 [NISTIR 8060] provides comprehensive guidance regarding the application of SWID tags, particularly as part of comprehensive software asset management lifecycles and cybersecurity procedures. [NISTIR 8060] highlights the stakeholder benefits that can be gained as SWID tags become more widely produced and consumed within the marketplace. That NISTIR also provides the following support for the application of SWID tags:

- Key SWID tag concepts that are helpful for understanding the different types of tags, how tags are created, and how tags are made available for use.

- A high-level overview of the SWID tag standard, explaining what a SWID tag is and how a tag encodes a variety of identifying and descriptive data elements about a software product.

- Implementation guidelines, including those for specific types of tags, that address common issues related to tag deployment and processing on information systems.

- Example usage scenarios regarding the use of SWID tags based on the SWID specification and NIST SWID tag guidance for software asset management and software integrity management.

NIST Internal Report 8085 is a complementary document, intended to assist with forming Common Platform Enumeration (CPE) names using the SWID structure.

CPE is a standardized method of naming classes of applications, operating systems, and hardware devices that may be present on computing devices. This report provides a model for using SWID tag data to create CPE names that conform to version 2.3 of the CPE Naming Specification [CPE23N]. Such CPE names are useful in support of numerous Software Asset Management (SAM) activities including software inventory, vulnerability management, and information security continuous monitoring. For example, because CPE names are used extensively in the National Vulnerability Database (NVD), SWID tag derived CPE names are useful to associate vulnerability reports with repository records of installed software products.

1.1 Purpose and Audience

This report provides guidance to enable the creation of CPE names using information extracted from SWID tags. By following the guidelines in this report, software asset managers and security professionals will be able to use specific elements and attributes of SWID tags to create accurate CPE 2.3 names. As the software community continues to expand the use of SWID tags, interoperability with existing CPE-based systems (e.g., NVD) will be enhanced.

The material herein addresses three distinct audiences. The first audience is *software providers*, the individuals and organizations that develop, license, and/or distribute commercial, open source, and custom software products. Software providers also include organizations that develop software solely for in-house use. The ability for software providers to easily create both SWID tags and CPE 2.3 names enhances software asset management and security continuous monitoring capabilities.

The second audience is *providers of inventory-based products and services*, the individuals and organizations that develop tools for discovering and managing software assets for any reason, including securing enterprise networks using information from standard inventory processes. This audience has unique needs because their products and services will consume and utilize information in both CPE names and SWID tags, as each becomes available on endpoints. The ability to create CPE names based upon information within SWID tags (e.g., software creator names, product names, product editions, software version information) enhances the ability of inventory-based products and services to achieve the cybersecurity goals described above.

The third audience is *software consumers*, the individuals and organizations that install and use commercial, open source, and/or in-house developed software products. This report helps consumers leverage CPE-capable products while gaining benefits of SWID tags as described in [NISTIR 8060]. Consumers are encouraged to request that software providers deliver products with SWID tags to achieve organizational software asset management and cybersecurity goals.

1.2 Document Structure

The remainder of this document is organized into the following sections and appendices:

- Section 2 provides guidance regarding forming Common Platform Enumeration (CPE) Names.
- Appendix A defines selected acronyms used in the document.
- Appendix B lists references that provide additional information or clarification.

2 Forming Common Platform Enumeration (CPE) Names

A component of NIST's Security Content Automation Protocol (SCAP), the Common Platform Enumeration (CPE) is a standardized method of naming classes of applications, operating systems, and hardware devices that may be present on computing devices.[1] NIST maintains a dictionary of CPE names as part of the National Vulnerability Database (NVD).[2] CPE names play an important role in the NVD, where they are used to associate vulnerability reports with the affected software products. Many cybersecurity products report discovered software using CPE names, and/or use CPE names to search the NVD for indications of software vulnerability. For these reasons, it is useful to specify a standardized, automatic procedure for forming CPE names using pertinent SWID tag attribute values. This section defines such a procedure.

The remainder of this section is organized as follows: Section 2.1 explains a number of challenges with forming CPE Names using information from a SWID tag. Section 2.2 provides an overview of the procedure to use in forming a CPE Name from a SWID tag. Section 2.3 details the procedure to use in forming a CPE Name from a SWID tag. Section 2.4 provides guidelines around the use of specific types of SWID tags to form CPE Names. Finally, Section 2.5 provides a summary of the information provided in this section.

2.1 CPE Name Forming Challenges and Solutions

The CPE Name Forming Procedure presented here conforms to version 2.3 of the CPE Naming Specification [CPE23N]. This specification defines eleven attributes comprising a well-formed CPE name (WFN):

- part
- vendor
- product
- version
- update
- edition
- language
- sw_edition
- target_sw
- target_hw
- other

[1] See: http://scap.nist.gov/specifications/cpe/.
[2] See: https://nvd.nist.gov/.

Two challenges must be addressed when forming a CPE name automatically from data contained in a SWID tag. The first challenge is *data insufficiency*, and the second is *non-ASCII characters*. These are discussed in the following subsections.

2.1.1 Data Insufficiency

A SWID tag that conforms only to the mandates and requirements set forth in the SWID specification would lack the data required to reliably populate nine of the eleven attributes of a CPE name. One could use `<SoftwareIdentity> @name` as the value for the CPE "product" attribute, and `<SoftwareIdentity> @version` as the CPE "version" attribute, but the other CPE attributes have no obvious sources within a SWID tag and thus would have to be left unspecified in any automatically generated CPE name.

Unfortunately, a CPE name that includes only a product name and a version will, in most cases, be insufficient for vulnerability management usage scenarios. In particular, using such a limited CPE name to search the NVD for vulnerability reports is likely to result in a *false negative*: a failure to discover relevant software vulnerability reports in the NVD even when such relevant reports exist. False negatives are likely because the SWID specification supplies only the `<SoftwareIdentity> @name` attribute to capture a product's market name, whereas the CPE specification breaks a product's name down into a set of fine-grained data elements, including *vendor, part, product, update, edition*/*sw_edition*, and *hw_edition*.

Consider a product with the market name assigned by the vendor of "Acme Roadrunner Home Edition Service Pack 2." This is the string that would be specified as the value of the product's `<SoftwareIdentity> @name` attribute in its primary tag. In contrast, a conventional CPE name as used within the NVD would break that string into the following CPE name elements:

vendor = "`acme`"
part = "`a`"
product = "`roadrunner`"
update = "`sp2`"
sw_edition = "`home`"

As a result, vulnerability reports in the NVD associated with "Acme Roadrunner Home Edition Service Pack 2" would be tagged with the following CPE standard-conformant name:

```
cpe:2.3:a:acme:roadrunner:*:sp2:*:*:*:home:*:*:*
```

Now consider attempting to generate a CPE name from the Acme Roadrunner product's primary SWID tag. A name generation procedure that used only the tag's `<SoftwareIdentity>` `@name` and `@version` attributes would produce the following CPE name (assuming straightforward replacement of whitespace with underscores, and character conversion to lowercase):

```
cpe:2.3:*:*:acme_roadrunner_home_edition_service_pack_2:*:*
:*:*:*:*:*:*:*
```

A search of the NVD using this generated CPE name—applying the matching algorithm that is defined as part of the CPE specification—would likely fail to find any records, including those records tagged with the standard-conformant name. This negative result would create the false impression that the Acme Roadrunner product is free of known vulnerabilities.

Guideline **PRI-13** in [NISTIR 8060] Section 5.2.4 requires that several additional data values be provided in SWID tags, using the `<Meta>` element:

- `@product`
- `@colloquialVersion`
- `@revision`
- `@edition`

In addition, guideline **GEN-3** in [NISTIR 8060] Section 4.3 requires authoritative tag creators to specify an `<Entity>` `@name` for the `softwareCreator` role, and guideline **GEN-4** encourages non-authoritative tag creators to do so whenever possible. These guidelines make it possible to form more useful CPE names from a SWID tag.

2.1.2 Non-ASCII Characters

CPE names are limited to the printable subset of the American Standard Code for Information Interchange (ASCII) character encoding set. In contrast, when strings are used as SWID tag attribute values, those strings may contain arbitrary Unicode characters. This creates a need for a standard approach for converting Unicode characters into ASCII characters acceptable within a CPE name.

IETF RFC 3490 on Internationalizing Domain Names in Applications (IDNA) [RFC 3490] offers a solution to this challenge. IDNA defines the concept of an ASCII-Compatible Encoding (ACE) of a string, which may contain arbitrary ASCII and non-ASCII characters, and further specifies a `ToASCII` procedure that converts such strings into strings composed of only ASCII characters. Although the output of `ToASCII` is not intended for human consumption, it provides a satisfactory encoding of the input that meets the requirements for CPE name attributes. In addition, IDNA also offers a `ToUnicode` procedure that takes an ACE string as input and reverses the encoding to produce an output string, which may contain Unicode characters. Consequently, guidance in this report will require that pertinent SWID tag attribute value strings are processed by an RFC 3490-conformant implementation of `ToASCII` during the CPE name forming procedure.

2.2 Overview of CPE Name Forming Procedure

The `CPENameGenerator` procedure, formally specified below, has the following steps:

1. Given an input SWID tag, a collection of *preliminary CPE name attributes* is extracted. These attributes are "preliminary" in the sense that their values are directly copied from the input tag and do not yet conform to the CPE attribute requirements (e.g., containing only printable ASCII characters).

2. Each preliminary CPE name attribute is converted to the ASCII encoding using the `ToASCII` procedure specified in [RFC 3490].

3. Any embedded whitespace characters are replaced with underscore characters.

4. Printable non-alphanumeric characters *except underscores* are quoted (i.e., a backslash character is inserted into the string immediately before the non-alphanumeric character.

5. Values for the final CPE name attributes are assigned. In most cases, final values are simply the results of the preceding four steps. Special conditions apply to how the CPE "product" value is assigned.

The `CPENameGenerator` produces a CPE WFN as its output. This WFN may then be bound to either a URI or a formatted string according to the `bind_to_URI()` and `bind_to_fs()` procedures specified in [CPE23N].

2.3 CPENameGenerator Procedure in Detail

The `CPENameGenerator` procedure is formally specified below.

2.3.1 Step 1 – Collect Preliminary CPE Name Attributes

Given an input SWID tag, extract the following preliminary attribute values:

```
prelimVendor := value of <Entity> @name where <Entity> @role contains
softwareCreator

prelimProduct := value of <Meta> @product

prelimProductDefault := value of <SoftwareIdentity> @name

prelimColloqVer := value of <Meta> @colloquialVersion

prelimVersion := value of <SoftwareIdentity> @version

prelimUpdate := value of <Meta> @revision

prelimEdition := value of <Meta> @edition
```

2.3.2 Step 2 – Convert to Pure ASCII

The `ToASCII` procedure is applied to each preliminary attribute value:

```
prelimVendor := ToASCII(prelimVendor)

prelimProduct := ToASCII(prelimProduct)

prelimProductDefault := ToASCII(prelimProductDefault)

prelimColloqVer := ToASCII(prelimColloqVer)
```

```
    prelimVersion := ToASCII(prelimVersion)

    prelimUpdate := ToASCII(prelimUpdate)

    prelimEdition := ToASCII(prelimEdition)
```

2.3.3 Step 3 – Replace Whitespace with Underscores

Apply the `eliminate_whitespace()` function to each preliminary attribute value:

```
    prelimVendor := eliminate_whitespace(prelimVendor)

    prelimProduct := eliminate_whitespace(prelimProduct)

    prelimProductDefault :=
      eliminate_whitespace(prelimProductDefault)

    prelimColloqVer := eliminate_whitespace(prelimColloqVer)

    prelimVersion := eliminate_whitespace(prelimVersion)

    prelimUpdate := eliminate_whitespace(prelimUpdate)

    prelimEdition := eliminate_whitespace(prelimEdition)
```

The `eliminate_whitespace()` function is defined as follows:

```
function eliminate_whitespace(s)
  ;; Inspect each character in string s. In the output, replace
  ;; any embedded whitespace characters with underscores.
  result := "".
  idx := 0.

  while (idx < strlen(s))
    do
      c := substr(s,idx,idx). ; get the idx’th character of s.
      if is_whitespace(c) then
        ;; Substitute an underscore for a whitespace character.
        result := strcat(result,”_”).
      else
        result := strcat(result,c).
      endif.
      idx := idx + 1.
  end.
  return result.
end.

function substr(s,b,e)
  ;; Returns a substring of s, beginning at the b’th character,
  ;; with 0 being the first character, and ending at the e’th
  ;; character. b must be <= e. Returns nil if b >= strlen(s).
end.

function strcat(s1,s2,...sn)
  ;; Returns a copy of the string s1 with the strings s2 to sn
  ;; appended in the order given.
  ;; Cf. the GNU C definition of strcat. This function shown
  ;; here differs only in that it can take a variable number
  ;; of arguments. This is really just shorthand for
  ;; strcat(s1, strcat(s2, strcat(s3, … ))).
end.

function strlen(s)
  ;; Defined as in GNU C, returns the length of string s.
  ;; Returns zero if the string is empty.
end.
```

2.3.4 Step 4 – Add Quoting as Required

Apply the `add_quoting()` function to each preliminary attribute value:

```
     prelimVendor := add_quoting(prelimVendor)

     prelimProduct := add_quoting(prelimProduct)

     prelimProductDefault := add_quoting(prelimProductDefault)

     prelimColloqVer := add_quoting(prelimColloqVer)

     prelimVersion := add_quoting(prelimVersion)

     prelimUpdate := add_quoting(prelimUpdate)

     prelimEdition := add_quoting(prelimEdition)
```

The `add_quoting()` function is defined as follows:

```
function add_quoting(s)
  ;; Inspect each character in string s. Alphanumeric characters
  ;; and underscores pass unchanged. All other characters are
  ;; prefixed with a backslash (quote) character.
  result := "".
  idx := 0.

  while (idx < strlen(s))
    do
      c := substr(s,idx,idx). ; get the idx’th character of s.
      if (is_alphanum(c) or c = "_") then
```

```
        ;; Alphanumerics and underscores pass untouched.
        result := strcat(result,c).
      else
        result := strcat(result,"\").
        result := strcat(result,c).
      endif.
      idx := idx + 1.
  end.
end.
```

2.3.5 Step 5 – Finalize the CPE WFN Attribute Values

The final CPE WFN attribute values are assigned as follows:

```
part := “*”

vendor := prelimVendor (if non-null) otherwise “*”

product := prelimProduct (if non-null) otherwise prelimProductDefault
  In addition, if prelimColloqVer is non-null, then add it to the product
  attribute:
    product := product + “_” + prelimColloqVer

version := prelimVersion

update := prelimUpdate (if non-null) otherwise “*”

edition := prelimEdition (if non-null) otherwise “*”

all other WFN attributes := “*”
```

The resulting eleven attribute values now satisfy the requirements of a CPE WFN and are suitable for binding to URI or formatted string names.

2.4 Guidelines on CPE Name Formation

This report concludes with guidelines related to the formation of CPE names from SWID tags.

The first guideline limits the applicability of CPE Name Formation to only two types of SWID tags: corpus and primary tags. Because corpus tags are used to describe software products in a pre-installation state, it is useful to be able to form CPE names from such tags in cases where CPE name information could be helpful in deciding, for example, whether to allow installation. Because primary tags describe software products installed on endpoints, it is useful to be able to form CPE names from such tags to support vulnerability management usage scenarios. Because CPE was never designed to support naming of patches, patch tags cannot be used as sources for CPE names. Supplemental tags are not useful as sources of CPE names since only corpus and primary tags may contain the necessary data values.

Guidelines on CPE name formation are provided as additions to the tag-specific implementation guidelines described in [NISTIR 8060] Section 5:

CPE-1. A corpus tag MAY be used as the source of a CPE name. When forming a CPE name from a corpus tag, the `CPENameGenerator` procedure MUST be followed.

CPE-2. A primary tag MAY be used as the source of a CPE name. When forming a CPE name from a primary tag, the `CPENameGenerator` procedure MUST be followed.

CPE-3. A patch tag MUST NOT be used as the source of a CPE name.

CPE-4. A supplemental tag MUST NOT be used as the source of a CPE name.

2.5 Summary

The above guidance provides a standardized, automatic procedure for forming CPE names using pertinent SWID tag attribute values. The ability to accomplish this automated formation depends upon the source SWID tag containing sufficient data to populate the CPE elements (e.g., vendor, part, product, update, edition/sw_edition, hw_edition.) Where such information is available within the applicable SWID tag(s), the CPE Name Forming Procedure described will help organizations to consistently achieve software asset management and security continuous monitoring objectives.

Appendix A—Acronyms

Selected acronyms and abbreviations used in this report are defined below.

ACE	ASCII-Compatible Encoding
ASCII	American Standard Code for Information Interchange
CPE	Common Platform Enumeration
CVE	Common Vulnerabilities and Exposures
IDNA	Internationalizing Domain Names in Applications
IEC	International Electrotechnical Commission
IETF	Internet Engineering Task Force
ISO	International Organization for Standardization
NIST	National Institute of Standards and Technology
NISTIR	National Institute of Standards and Technology Internal Report
NVD	National Vulnerability Database
RFC	Request for Comments
SAM	Software Asset Management
SCAP	Security Content Automation Protocol
SWID	Software Identification
URI	Uniform Resource Identifier
WFN	Well-Formed CPE Name

Appendix B—References

[CPE23N] Cheikes, B. A., Waltermire, D., and Scarfone, K. *Common Platform Enumeration: Naming Specification 2.3*. National Institute of Standards and Technology Interagency Report 7695, August 2011. http://csrc.nist.gov/publications/nistir/ir7695/NISTIR-7695-CPE-Naming.pdf [accessed 8/26/15].

[ISO/IEC 19770-2:2015] International Organization for Standardization/International Electrotechnical Commission, *Information technology -- Software asset management -- Part 2: Software identification tag*, ISO/IEC 19770-2:2015, 2015. http://www.iso.org/iso/catalogue_detail?csnumber=65666 [accessed 12/11/15].

[ISO/IEC 19770-5:2013] International Organization for Standardization/International Electrotechnical Commission, *Information technology -- Software asset management -- Part 5: Overview and vocabulary*, ISO/IEC 19770-5:2013, 2013. https://www.iso.org/obp/ui/#iso:std:iso-iec:19770:-5:ed-1:v1:en [accessed 8/26/15].

[NISTIR 8060] Cheikes, B., Feldman, L., Waltermire, D. & Witte, G. (2015), *Guidelines for the Creation of Interoperable Software Identification (SWID) Tags*. National Institute of Standards and Technology Interagency Report 8060, December 2015. http://csrc.nist.gov/publications/PubsNISTIRs.html#NISTIR8060 [accessed 12/17/15].

[RFC 3490] Faltstrom, P., Hoffman, P., and Costello, A. (2003). *Internationalizing Domain Names in Applications (IDNA).* Internet Engineering Task Force (IETF) Network Working Group Request for Comments (RFC) 3490, March 2003. https://www.ietf.org/rfc/rfc3490.txt [accessed 8/26/15].