CPE Developer Days

April 30, 2008

Table of Contents

Introduction

CPE Developer Days was held on April 30th, 2008 at The MITRE Corporation in Bedford, MA. The purpose of this event was to discuss, in technical detail, the more difficult issues facing CPE, and to help drive future changes / enhancements to the enumeration. By bringing together the lead proponents within the CPE Community, the hope was to derive solutions that would benefit all parties, and help continue the ongoing development of the enumeration. What follows is a detailed summary of the discussions from this event.

In addition to a summary of the discussions, a list of action items has been recorded at the end of this document. These items represent things that were flagged as needing further discussion, or things that need to be further worked based on the discussions had.

Attendee List

Assuria Ltd	-	Chris Wood
Belarc	-	Richard DeFuria
	-	Gary Newman
DoD	-	Vladimir Giszpenc
	-	Lt Col Joseph L. Wolfkiel
McAfee	-	Kent Landfield
	-	David Raphael
MITRE	-	Jon Baker
	-	Steve Christey
	-	David Mann
	-	Bryan Worrell
	-	Margie Zuk
nCircle	-	Jay Graver
	-	Will Weisser
NIST	-	John Banghart
Palamida	-	Ernest Park
Redseal Systems	-	Joshua Keich
Secure Elements	-	Scott Carpenter
	-	Sudhir Gandhe
Sun Microsystems	-	John Totah
CPE Team	-	Andrew Buttner
	-	David Waltermire
	-	Harold Booth

Day One

Use Cases

Before diving into issues with the current specification and thinking about possible enhancements, it was important that everyone understood the current use cases for CPE. This discussion hopefully helped frame the environment that we are all working in and gave us context as we discussed individual issues. If the discussion led us to additional use cases that should be outlined in the specification, then that would be another worthwhile outcome of our conversation.

It was noted at the start that MITRE could do a better job expressing the different use cases that drive CPE. Use cases are fundamental to understanding the problems trying to be solved and a better understanding of these use cases will help the community work toward possible solutions. MITRE will take on this challenge and attempt to expand the use case section of the specification.

There are three current use cases in the specification and each were summarized at this time.

Common Names

The first use case defined for CPE is referred to as "Common Names". The basis of this is user-to-user, or tool-to-tool communications. In order for entities to exchange information about a platform, each entity must use a common name to express the platform in question. Otherwise the different entities will not understand what it is they are saying. This is the classic 'windows 2000' / 'win 2k' issue.

This is the same use case that drives other enumerations like CVE and CCE. The sharing of information requires a common name for the things being shared so that both parties can understand the information. This use case was actually the main driver behind CPE and was first realized at an OVAL Compatibility testing session as two tools were trying to share results of an evaluation based on a certain platform, but were unable to understand the platform names stated by the other tool, and hence did not present all the results that it should.

Matching

This use case has its originates from the XCCDF team as they realized a need to not only have common names, but also be able to talk at different levels of abstraction and make sense of it all. Other enumerations often focus on a concrete thing (like a vulnerability) and the enumeration simply needs to provide a common name for that thing. But when dealing with platforms, the concrete thing we are talking about is often correctly referred to by many different possible names. For example, a system may be both 'windows' as well as 'windows xp sp1'. The later name is just a more descriptive name, provided at a different level of abstraction. CPE needs to understand this and provide a structure for understanding the common name when dealing at different levels of abstraction.

Inclusive / Exclusive

It was asked if this use case deals with the notion of whether a given CPE Name is inclusive or exclusive. In other words, is it the matching use case that lets us know if the intention of a given CPE Name is to include all future platforms that correspond to the name, or if the intention is to exclude all future platforms? This asks the question: "Are there different types of matching?"

In this case, it is the use case description that is probably leading to the question above. Matching is more of a function than a use case. Use cases in general describe communication between parties, but matching is more focused on how to solve this communication. We need to redo this use case and describe it in a different way to help better understand the need(s) we are dealing with.

Regarding the inclusive/exclusive discussion, it was stated we need to develop some best practices so that users know how to use CPE Name correctly. What type of clarifying information needs to be presented when using a CPE Name to help a user understand how to digest the information? This type of clarifying information is important, but is outside the scope of CPE. CPE focuses on identifying and naming the different buckets of platforms, relationships to these buckets are out of scope. Having said that, it is important that CPE educates the user about this distinction and a best practices document would be a start in the correct direction.

Ranges

Another tangential discussion that came up was related to the need for the ability to specify ranges in a CPE Name. The desire here is to use a name that, for example, identifies all versions of a product prior to some number. The lack of range specification is considered by some to be a weakness in CPE. It is often impossible to enumerate all versions of a particular product and having the ability to specify ranges would help align CPE with real world uses. Note that in some cases, CPE can handle ranges already. Remember that a CPE Name enumerates a 'bucket' of platforms. For example, cpe:/o:vendor:product:3 would identify all version 3 releases (3.0, 3.1, 3.2, ...) of the given product. The CPE Language might also be able to be used and/or expanded to help with the 'or earlier' problem. MITRE will further explore the issue of ranges and clarify / modify the specification as necessary.

Prefix Property

The previous discussion on ranges brought up the point that platforms often don't follow a tree / hierarchy (making ranges difficult), yet CPE is based on tree due to the prefix property as outlined in the requirements of the naming format. Should CPE remove the prefix property requirement? Historically, it was the matching use case that brought about the prefix property. We needed a way to express more information inside the identifier so that different level of abstractions could be worked with. If we remove the matching use case, we could remove the prefix property.

Alternatively, we could push the matching function down to the tool level. In other words, if a tool was handed a name for 'windows xp' and for 'windows xp sp1', the tool itself would be responsible for determining the set of individual platforms that belong to each name and then determine that one name matches the other. The prefix property just allows a tool to do this matching without having to go

through the above logic. This discussion was tabled until we have a better understanding of the different use cases.

Reporting

The reporting use case focuses on the need to cross reference data found in different locations. For example, a report may be generated about existing systems identified on a given network. If a CPE Name is provided for each system, that identifier can be used to research a 3rd party site about possible vulnerabilities that are applicable for that type of platform. This use case is obviously very similar to the Common Name use case.

Discussion about this specific use case did not occur due to time constraints, but further exploration of this will occur as MITRE looks to expand and improve the use case section of the specification.

Additional Use Cases

An additional use case was proposed and discussed. It focused around the ability to find the CPE Name desired by a user.

Finding

How do you (as a human) find the CPE Name you are looking for? Similar to this, how do you take the names in the Official CPE Dictionary and match them up to the names in a proprietary listing of names? For example, a user might be trying to find the CPE Name for Windows Server 2003 in order to look up information about that platform. How can the user find the correct name for the platform? We can't expect users to read and understand the specification and we can't expect them to know all the rules associated with the creation of names.

It was pointed out that the user shouldn't have to be dealing with CPE Names at all. The tool they are using to search for information should abstract out the complexities of CPE and present the user with something they are more familiar with. There are many different ways that an interface can be created to help this problem. It is this interface that is responsible for mapping the mechanisms being presented to the user with the corresponding CPE Names.

So how does a tool perform this matching? CPE does its best to support this by formalizing the naming specification and making an attempt to have names created in a predictable way. Maybe expanding the metadata associated with a given CPE Name can help in this process? But at the same time is this all something that really should be built on top of CPE, and not be part of CPE?

This topic will be looked at further as a possible use case, and ways to support the need will be explored by the dictionary. It is obvious that users of CPE need an available resource to support their mapping needs.

Official CPE Dictionary

The second discussion of the day was used to present an update on the status of the Official CPE Dictionary and to talk about some of the issues facing its implementation. The Official CPE Dictionary is

hosted at NIST and they have been working on standing up the infrastructure and processes to support it. On April 15th 2008 an updated version of the dictionary was made available that included over 250 vendors and over 15,000 individual names. NIST is continuing to work on and improve this dictionary and presented these plans to the attendees at CPE Developer Days.

One area that is currently lacking in the dictionary is related to OVAL Definitions. There are only 58 CPE Names that are mapped to an OVAL Definition. This really limits the power of the dictionary and hopefully we can work to improve this. This should be one of the things we all push for as me move forward.

Issues / Decisions

NIST stepped through a variety of issues (related to creating valid CPE Names) that they have dealt with over the past few months, and discussed how they dealt with each issue. A few of these issues are still open and the audience was engaged in discussion about these.

- When presented with software that is known by both a marketing version and a product version, the product version was used as this allows correlation between multiple versions. The specification needs to clarify this and state the desired choice when presented with this type of challenge.

- What platform part should firmware be associated with? (hardware, os, or application) For now firmware will be associated with hardware. In the future, the addition of a 'firmware' part would help solve this issue.

- It is a bit unclear which name to create when dealing with product families. For example, should a CPE Name be created for the Cisco 2600 Series or for Microsoft Office? These types of examples are not real products, but rather are either families of related products or bundles of individual products. It was noted during discussion that the specification does not limit naming to one particular type and names for both the families/bundles and the individual products are both acceptable.

- CPE Names for initial software releases need to have a unique name. The CPE Specification needs to state a term (such as GA, RTM, Gold, etc.) to use in the version component for this case.

Interfaces

The next topic of conversation was related to of the accessibility of the Official CPE Dictionary. Currently the dictionary is only offered as a single XML file for download off the web site. It is acknowledged that this is not ideal (especially given the size of the file) and the goal is to develop interfaces for users to request subsets of the dictionary related to their needs. This will hopefully be accomplished through a web based search interface that allows the user to perform a custom search and then download just the results returned. Additionally, a set of web services are also planned to support automated retrieval of CPE Names from the dictionary.

One question asked was about the ability to grab a 'what has changed' file. This is something that should be able to be handled via the web services. Using the metadata included in the dictionary, and working with the proposed lifecycle model (discussed below), a user should be able to construct a query to grab just those CPE Names that are new since the last download.

The WSDL will be published to the CPE Community when it is available. Drafts of the WSDL will be shared during development so the community can comment as necessary. Please pay attention to the CPE Discussion list for this.

Content Explanation

NIST will be including a few pieces of additional metadata along with the CPE Name to help support additional use cases. One such addition is a hierarchical component tree that presents unique components and an associated title. This will help those users that need to present through an interface a way to build a CPE Name from its pieces. For example, instead of presenting the actual component name (which might be confusing to a reader), a well-known title for that component can be presented.

Lifecycle information will also be available through metadata. The lifecycle process itself will be discussed below. The metadata to support this includes an NVD id. This is a numerical id that allows NIST to track a CPE Name through the lifecycle process. The need for this id is due to the fact that a CPE Name may change during its draft phase. Please note that the NVD id is only meant for internal use and should not be used by the community.

One other point that was made was that where appropriate only 'leaf nodes' are included. What is meant by leaf nodes is a CPE Name that includes all the components in the name, as opposed to names that only include some components. The reason for this is to control the size of the dictionary at this time. From a leaf node, any of the other smaller names can be obtained. There are some cases where individual titles or references are needed for these smaller names and those will be included as necessary.

Discussion surrounding the above topic focused on the fact that this approach could be problematic when dealing with languages as there will be a large number of 'leaf nodes' that will be tough to fully enumerate. It was clarified that the intent is not to fully enumerate leaf nodes, but rather to avoid providing all the smaller names (names with less components specified) when the only the larger name is used by the community. This is a topic that needs to have some more discussion once experience is gained.

Lifecycle Process

For this part of the discussion, NIST presented a proposed lifecycle process to support the maintenance of CPE Names in the dictionary. The proposed process involves four stages: new, draft, final, retired. At the new stage, the CPE Name is subject to change as it is reviewed / normalized by the dictionary authority. At the draft stage, the CPE Name appears in the dictionary but the name is not yet ready for prime time. CPE Names in the draft stage are available for public review and comment. If a name in the draft stage does not change for 30 days then the name is moved to the final stage, where it is frozen and can no longer change. CPE Names in the final stage can be deprecated as necessary, but they remain in

the final stage. Once a CPE Name has been deprecated for one year, the name then moves into the retired stage. At this stage, the CPE Name is removed from the Official CPE Dictionary, although it remains in NIST's repository for historical purposes.

A question was raised regarding vendor submitted names. The proposal has these names going through the same process as community submitted names. But should vendor names be given more trust in an attempt to speed up the process for these? Also, the vendor may not want the community to change their 'official' names. It was agreed that some validation must be made on vendor submitted names to make sure that they are creating names correctly. So the question is more accurately stated: Should vendor submitted names go through the draft stage? The conclusion was that we wanted to have only one process to manage and having every name follow the same step simplifies things.

NVD ID

Further discussion was had regarding the internal NVD id. This numerical id is created by NIST to support the lifecycle process and is exposed through the metadata found in the Official CPE Dictionary. One concern about this is that by exposing this id, it will be hard to keep people from using it and if they do we will have two different ids. What about not exposing this id? One argument for its inclusion was that without it, a change log would need to be created to expose a name's progression through the lifecycle process. It was decided that we need to have more discussion about the different use cases in order to make the correct decision here.

Working Session 1

The first working session focused on a number of topics that have been proposed as changes to CPE to help resolve specific deficiencies. For each topic we discussed the root problem as well as if the proposed solution would help solve it.

Aliases

The idea behind aliases is to link related CPE together. Basically to provide some mechanism that allows a user to find other CPE Names that might be of interest given the current name they are looking at. Maybe this could help solve the suite / bundle of products issues that was brought up during the Official CPE Dictionary discussion. Maybe this could help solve the issue related to changes in vendor / product names. But is this proposal opening us up to a maintenance headache as we would now have to figure out a way to manage these relationships.

Three different notions of aliases were discussed: 1- relating similar CPE Names to each other, 2- relating different types of information to a CPE Name, 3- relating multiple titles for a given CPE Name.

In the real world, products do have multiple names. Aliases give us the ability to link these names together. But the idea of linking the same names together falls apart in that there should be only one CPE Name for each product. If we end up with multiple names for the same product then we are not following the CPE Specification.

Rather than linking CPE Names that mean the same thing, maybe what we really need is a way to link the different test conditions that can be used to verify a CPE Name. For example, checking the version of winword.exe, evaluating a given OVAL Definition, or some other check might all be related to the cpe:/a:Microsoft:word CPE Name. This type of information is needed by the community and is only possible if we have the common name to link to. What is really desired here is a repository of information (relationships) that leverages CPE to create these links.

After further discussion, a conclusion was reached that the true need for aliases is related to titles. Users often know a platform by different names, and if they need to determine which CPE Name is related to their platform, then need to be able to use this name to find it. This is related to the proposed 'finding' use case that we talked about earlier. For example, cpe:/o:Microsoft:windows-xp should be associated with both "Microsoft Windows XP" and "winxp".

We were reminded that the real need of aliasing was to relate similar (not the same) CPE Names. For example, products that have changed names or vendors that switched versioning mechanisms need a mechanism to provide a link between the two. This is similar to the current deprecation feature. A possible solution is to provide an 'aliased_with' attribute in the same way.

Wildcards

One of the biggest known issues with the current CPE Specification is that the version component does not support matching. In addition, the lack of any logic makes it impossible to create names that represent a range of versions. The proposal is to introduce certain wildcards that would allow CPE Names to be created that represented a desired range of versions.

For example, think of the component "3.*" as representing "any valid minor release of version 3". The alternative way (and current way as defined by the specification) to create this type of name is to just have a component value of "3". The issue though is that through matching, there is no way to know that a name with a component value of "3.2" would match that name with a component value of "3". By using a wildcard, matching could be improved so we could know that "3.2" matches "3.*". The hope is that this would provide more flexibility in the ability to request information via CPE.

The dictionary actually already has implemented this feature and is testing it to see how viable it is. The question is whether this capability should be part of the CPE Specification or if this is something that is provided as a search interface to the Official CPE Dictionary as a helpful feature.

An area that this type of feature would benefit is when a CPE Name is included as a reference in an external entity. For example, a CVE entry can use a CPE Name with a wildcard to refer to the affected platforms for that vulnerability. In this case, the wildcard feature would indeed be part of the specification.

Versions

One thing that makes wildcarding difficult is the variety of version structures found within the industry. It is hard, if not impossible, to find a consistent way to represent versions, thus making a wildcarding format also difficult.

Another proposed feature that might be able to be leveraged here is tagging. Each CPE Name could be tagged with certain version information specific to the name being created. This has the benefit of allowing those who create the CPE Name (and thus know the version structure) to add the information at that time.

These tags could then be used in a search to find the CPE Name that is desired. Unfortunately, to leverage this type of search, we would have to change the way we reference platform information. Instead of just using a CPE Name (unique identifier), we would now need some type of structure / language to do the job.

Matching

It was asked if matching should be performed on the names themselves. Based on experience, matching is best done using more than just the identifier. Put another way, matching often requires more information that is not encoded in the CPE Name. A lot of the discussion today has been about fitting more information into the identifier to support these types of functions. Should we instead just remove the notion that matching should done only via the CPE Name itself? This is something that requires further discussion.

Abbreviations

The use of abbreviations has come under fire in the past. The goal of abbreviation is to condense certain terms that are commonly known by a shorter form. (e.g. “ie” instead of “internet explorer”)

It was noted that reducing size and conserving bandwidth does not seem to factor into any of the other decisions and discussions had throughout the community. So it was asked why would it factor into the discussion about abbreviations? This feature just complicates things and makes the creation of names more difficult.

One place where abbreviations do help is in formalizing the creation of new names. There are many terms / product names that are known by both a full name and an abbreviation, many where the abbreviation is actually the only name known. (e.g. “cisco ios” instead of “cisco internetwork operating system”) Having a table of known abbreviations (as currently in the specification) and stating that one should use the abbreviation instead of the full name helps writers know which term to use when creating a new CPE Name.

Major Version Transition

Due to time constraints, the discussion about challenges we will face during the next major version transition did not occur. However, it has been agreed that CPE should not push for a new major version at this time and should instead focus on better understanding the current 2.x version. There are many

things that we have discussed that can be rolled into the current spec as a minor version. Because of this, we can postpone this discussion until a later date.

Working Session 2

The second working session focused on three additional topics that have been proposed in the past by the CPE Community.

CPE Coverage

Currently CPE defines three different parts: hardware, operating system, and application. There have been proposals to add additional parts to CPE to help create names for other platform types. One of the suggestions that was discussed earlier is firmware. It would be simple to add 'f' as a valid part component value. This would not change any existing CPE Names.

One point to clarify is that each part type needs to be clearly described; what falls under each type, and when each type should be used. Drivers and libraries might be good possibilities for addition, but what exactly is meant by each of those terms? What is a driver? What is a library? Before adding in new parts like those, each must be defined and understood.

Removing the Platform Part

There was some discussion about removing the platform part all together as it can be confusing to some users and the rest of the CPE is usually unique anyway so the part is not really serving any purpose. It was pointed out that some hardware and custom operating systems are often referred to by the same names and so the platform part is in fact the piece that creates uniqueness. In addition, there are cases where a user may want to define a platform type of "all operating systems from Cisco". In a case like this (where the vendor develops both hardware and software) the platform part does add value to the CPE Name format.

It was also proposed that the platform part could be moved to a different place in the URI. But it was realized that this really doesn't change the types of names that can be created (since the blank component is an option). It doesn't matter if the platform part comes before or after vendor, and it is probably best in this case to leave it as is and not create new changes.

Numerical Identifiers

A number of community members have brought up the idea of using a numerical identifier (as opposed to the current URI string) for CPE. There are a number of advantages with a numerical identifier. Maybe the biggest is that they are more stable in an environment that sees a lot of changes (vendor and product names). In addition, numerical identifiers are simpler since they do not encode any additional information. But this simplicity is also its biggest disadvantage as the ability to do matching with just the ids is lost. Instead, there would be a need to reference additional data sources to obtain the required information.

The URI structure is nice in that it does provide some amount of information in the id. Information that is dynamically accessed makes using common names more difficult to deal with. The URI is great for the

defined use cases and should not be removed, but adding on some type of dynamic information might help support additional use cases that are currently left out in the cold.

The internal NVD id (metadata that is found in the Official CPE Dictionary) was discussed again as this gives us a way to try adding a numerical identifier on a temporary basis to see if it can work, and also would make a transition in the future easier. We have to be careful though as providing multiple identifiers goes against our desire for a single common identifier. In reality, some people will use the NVD id instead of the CPE Name if we provide it. Enabling this will prevent us from truly understanding the strengths and weaknesses of the current CPE Name structure.

Using a Tagged Approach

Tagging has been proposed as a way to not only breakdown the tree like structure of CPE, but also as a way of giving those creating CPE Names more flexibility in the information that they are providing. One approach to this proposal is to have the CPE Name be a numerical identifier and then use tags to supply information that would be needed to support matching. In a way, treat a CPE Name similar to a record in a database. This would give the added benefit of being able to change the value of certain tags (e.g. vendor name) without changing the CPE Name itself.

The approach above makes interacting with CPE very similar to interacting with a database. It was noted that for every use so far, a copy of the Official CPE Dictionary will need to be carried around by the user anyway, so requiring a tool to reference this file/database would not be that challenging.

Incorporating tags does not have to be such a radical change. The ability to tag something could be built around the current URI based CPE Name. The tags in this case would be used to add additional information.

A great use of tags would be for relating CPE Names that are related in the function they serve. For example, names for apache and IIS could be tagged with 'web server'. This would greatly enhance the searching that could be done. Similarly names for IIS could be tagged with 'Internet Information Services'

Some tags need to be mandatory and defined. Information like vendor and product name need to be consistent and these should not be tags, but rather fields. Tags should be left for optional, arbitrary data.

A possible implementation of tags would be to have a namespace for the tag to identify who added the tag, a name for the tag, and finally a value for the tag. But is contributor really important? The answer to this depends on the approval approach that is adopted. If we have a defined lifecycle process like we talked about earlier, then a namespace for the tag is not important. But if we chose some wiki type approach, then a namespace becomes necessary.

Based on community experience, it was noted that there are really two types of tags. There are free-form tags which really only end up being useful to human readers, as they are the only ones able to sort

through the inconsistency. Then there are more formalized tags also known as categories, which really help support automation. CPE could use both types.

Probably the biggest advantage of a tagging approach is that new tags can be added on the fly instead of having to go back to a standards body and wait for them to be added to the specification. Arbitrary tags give the users more flexibility going forward. Also, those tags that become important will 'rise to the top' and the community could then go to the standards body to ask that those tags be added to the specification.

CPE Compatibility

Due to time constraints, the discussion about a possible CPE Compatibility Program was not held. If desired, this discussion can be had at a future meeting.

Wrap-Up

One final point of discussion involved the direction and goals of CPE. It was noted that we should look at CVE (Common Vulnerabilities and Exposures) and examine why that program has become a mature and successful standard. There are many similarities between the goals of CVE and the goals of CPE. Both have at their core the desire to enumerate a 'thing' and provide a unique common identifier. In CVE's case, they stayed focused on this goal and did not try to expand the scope to accommodate additional use cases. Instead, CVE let others (e.g. NVD) build on top of the CVE Dictionary in order to add functionality that was needed by specific users.

CPE has started down the opposite path with the enumeration trying to support a number of different use cases and building into its dictionary features needed by certain implementations. It was noted that the CPE Dictionary is acting more as a "National Software Registry" than as a dictionary of CPE Names. The suggestion was that the CPE Community might be able to learn from CVE's success which has been based on staying minimalistic and making sure it succeeds at that simplified task in order to support broad and currently unexpected usage.

One of the major challenges within the community is that almost everyone has their own needs that they want CPE to fill. It is impossible to not let these needs affect the discussion. Having these needs is fine, and it actually helps everyone understand the different use cases out there, but trying to satisfy all these (diverse) needs with one standard is a huge challenge and has slowed progress at times.

The question for the community is should CPE retract some of its use cases and instead focus on a much simpler goal? A lot of the discussion during the day was related to how best to add additional metadata. It had nothing to do with providing unique identifiers. The problem is that each use case has a different need for metadata. If CPE were to focus on just providing unique identifiers, and let other implementations supply the necessary metadata as needed, then we as a community would have a better chance of succeeding with the first goal while enabling others to address more specific use cases.

To accomplish the above, one option presented was to split the current Official CPE Dictionary into two separate implementations. One would be a simplified dictionary with little to no metadata. In essence, it would become something similar to the current CVE dictionary. The other piece would be a database of information about platforms that is keyed using a CPE Name. In a way, this could be something similar to the current National Vulnerability Database. This database would be much easier to create and evolve if it was not concerned with the overhead of a standard.

Action Items

- expanded/improved use case descriptions
- best practices document about how to use CPE correctly
- explore improving the format of CPE Names to include ranges
- discuss an additional possible use case of “finding”
- look to add additional metadata to each name to support developers mapping to internal names
- increase the number of CPE Names that are mapped to an OVAL Definition
- update specification to clarify issues raised by NIST while working on the Official CPE Dictionary
- publish draft of WSDL related to the Official CPE Dictionary
- explore the possible addition of an ‘aliased_with’ attribute
- further discussion about the approach to matching
- updated specification to include firmware
- further discussion about the possible addition of drivers and libraries as valid parts
- provide the ability to relate multiple keywords (known titles) to a given CPE Name
- explore the possibility of adding an optional tagging structure to the current CPE Specification
- explore the idea of simplifying the current dictionary and relying on external repositories for additional metadata