Vulnerabilities & PILAR (versión 7.3)

2.7.2019

1 Introduction

1.1 CPE – Common Platform Enumeration

Common Platform Enumeration (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.

The CPE Product Dictionary provides an agreed upon list of official CPE names. The dictionary is provided in XML format and is available to the general public. The CPE Dictionary is hosted and maintained at NIST, may be used by nongovernmental organizations on a voluntary basis, and is not subject to copyright in the United States.

https://nvd.nist.gov/products/cpe

Example:

```
  <cpe-item name="cpe:/a:microsoft:active_directory_federation_services:2.1">
    <title xml:lang="en-US">
      Microsoft Active Directory Federation Services 2.1
    </title>
    <references>
      <reference
        href="http://technet.microsoft.com/en-us/windowsserver/dd448613.aspx">
        Vendor website
      </reference>
    </references>
    <cpe-23:cpe23-item
name="cpe:2.3:a:microsoft:active_directory_federation_services:2.1:*:*:*:*:*:*:*"
/>
  </cpe-item>
```

1.2 CVE – Common Vulnerabilities and Exposures

The Common Vulnerabilities and Exposures (CVE) system provides a reference-method for publicly known information-security vulnerabilities and exposures. The National Cybersecurity FFRDC, operated by the Mitre Corporation, maintains the system, with funding from the National Cyber Security Division of the United States Department of Homeland Security.

https://cve.mitre.org/

Example:

```
<?xml version='1.0' encoding='UTF-8'?>
<nvd
  pub_date="2019-01-09T03:00:03" ...>
```

```
  <entry id="CVE-2018-0001">
    <vuln:vulnerable-software-list>
      <vuln:product>cpe:/o:juniper:junos:12.1x46:d10</vuln:product>
      <vuln:product>cpe:/o:juniper:junos:12.1x46:d15</vuln:product>
      <vuln:product>cpe:/o:juniper:junos:12.1x46:d20</vuln:product>
    </vuln:vulnerable-software-list>
    <vuln:cve-id>CVE-2018-0001</vuln:cve-id>
    <vuln:published-datetime>2018-01-10T17:29:00.930-05:00</vuln:published-
datetime>
    <vuln:last-modified-datetime>2018-02-22T21:29:02.140-05:00</vuln:last-
modified-datetime>
    <vuln:cvss>
      <cvss:base_metrics>
        <cvss:score>7.5</cvss:score>
        <cvss:access-vector>NETWORK</cvss:access-vector>
        <cvss:access-complexity>LOW</cvss:access-complexity>
        <cvss:authentication>NONE</cvss:authentication>
        <cvss:confidentiality-impact>PARTIAL</cvss:confidentiality-impact>
        <cvss:integrity-impact>PARTIAL</cvss:integrity-impact>
        <cvss:availability-impact>PARTIAL</cvss:availability-impact>
        <cvss:source>http://nvd.nist.gov</cvss:source>
        <cvss:generated-on-datetime>2018-01-30T17:21:59.327-
05:00</cvss:generated-on-datetime>
      </cvss:base_metrics>
    </vuln:cvss>
    <vuln:cwe id="CWE-416"/>
    <vuln:references xml:lang="en" reference_type="UNKNOWN">
      <vuln:source>BID</vuln:source>
      <vuln:reference href="http://www.securityfocus.com/bid/103092"
xml:lang="en">103092</vuln:reference>
    </vuln:references>
    <vuln:summary>A remote, unauthenticated attacker may be able to execute code
by exploiting a use-after-free defect found in older versions of PHP through
injection of crafted data via specific PHP URLs within the context of the J-Web
process. Affected releases are Juniper Networks Junos OS: 12.1X46 versions prior
to 12.1X46-D67; 12.3 versions prior to 12.3R12-S5; 12.3X48 versions prior to
12.3X48-D35; 14.1 versions prior to 14.1R8-S5, 14.1R9; 14.1X53 versions prior to
14.1X53-D44, 14.1X53-D50; 14.2 versions prior to 14.2R7-S7, 14.2R8; 15.1 versions
prior to 15.1R3; 15.1X49 versions prior to 15.1X49-D30; 15.1X53 versions prior to
15.1X53-D70.</vuln:summary>
  </entry>
</nvd>
```

1.3 CVSS – Common Vulnerability Scoring System

The **Common Vulnerability Scoring System** (**CVSS**) is a free and open industry standard for assessing the severity of computer system security vulnerabilities. CVSS attempts to assign severity scores to vulnerabilities, allowing responders to prioritize responses and

resources according to threat. Scores are calculated based on a formula that depends on several metrics that approximate ease of exploit and the impact of exploit. Scores range from 0 to 10, with 10 being the most severe. While many utilize only the CVSS Base score for determining severity, temporal and environmental scores also exist, to factor in availability of mitigations and how widespread vulnerable systems are within an organization, respectively.

The current version of CVSS (CVSSv3.0) was released in June 2015. Version 3.1 is expected in 2019.

Example (xml): see in CVE example

Example (json):


```
  "impact" : {
    "baseMetricV3" : {
      "cvssV3" : {
        "version" : "3.0",
        "vectorString" : "CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H",
        "attackVector" : "NETWORK",
        "attackComplexity" : "LOW",
        "privilegesRequired" : "NONE",
        "userInteraction" : "NONE",
        "scope" : "UNCHANGED",
        "confidentialityImpact" : "HIGH",
        "integrityImpact" : "HIGH",
        "availabilityImpact" : "HIGH",
        "baseScore" : 9.8,
        "baseSeverity" : "CRITICAL"
      },
      "exploitabilityScore" : 3.9,
      "impactScore" : 5.9
    },
    "baseMetricV2" : {
      "cvssV2" : {
        "version" : "2.0",
        "vectorString" : "AV:N/AC:L/Au:N/C:P/I:P/A:P",
        "accessVector" : "NETWORK",
        "accessComplexity" : "LOW",
        "authentication" : "NONE",
        "confidentialityImpact" : "PARTIAL",
        "integrityImpact" : "PARTIAL",
        "availabilityImpact" : "PARTIAL",
        "baseScore" : 7.5
      },
      "severity" : "HIGH",
      "exploitabilityScore" : 10.0,
      "impactScore" : 6.4,
```

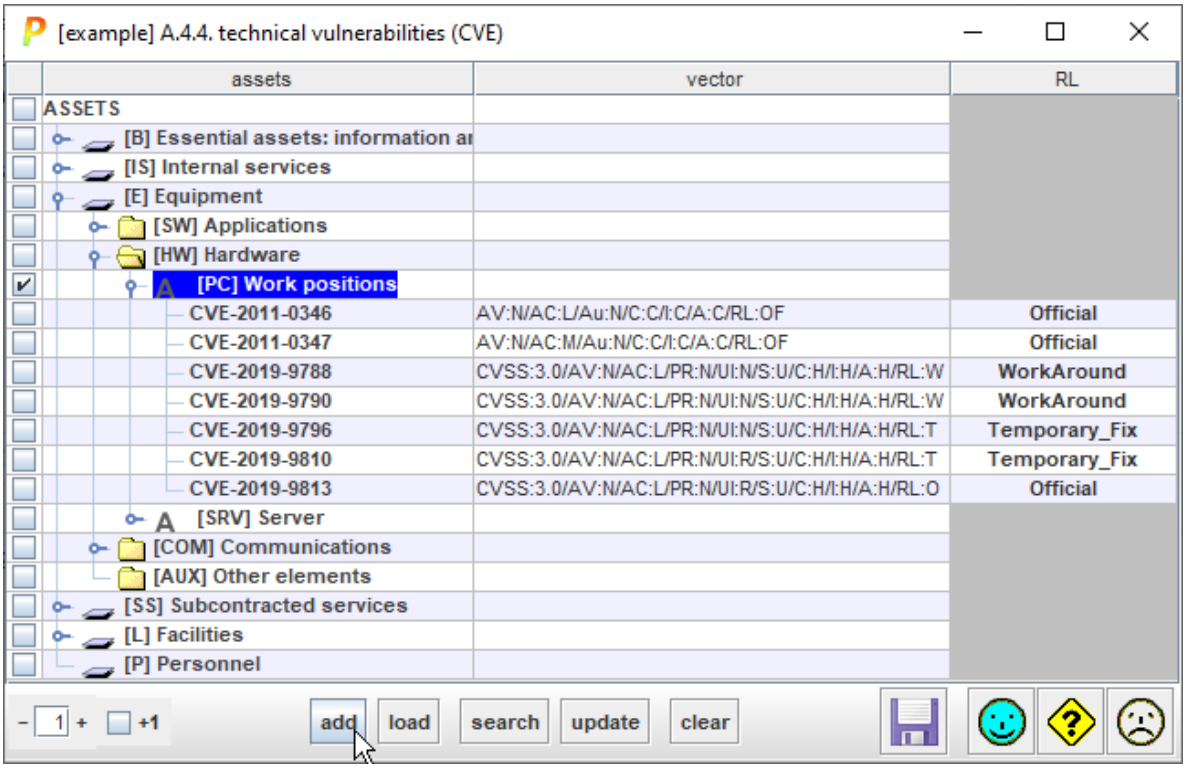
```
      "obtainAllPrivilege" : false,
      "obtainUserPrivilege" : false,
      "obtainOtherPrivilege" : false,
      "userInteractionRequired" : false
    }
  },
```

2 Load CVEs in PILAR

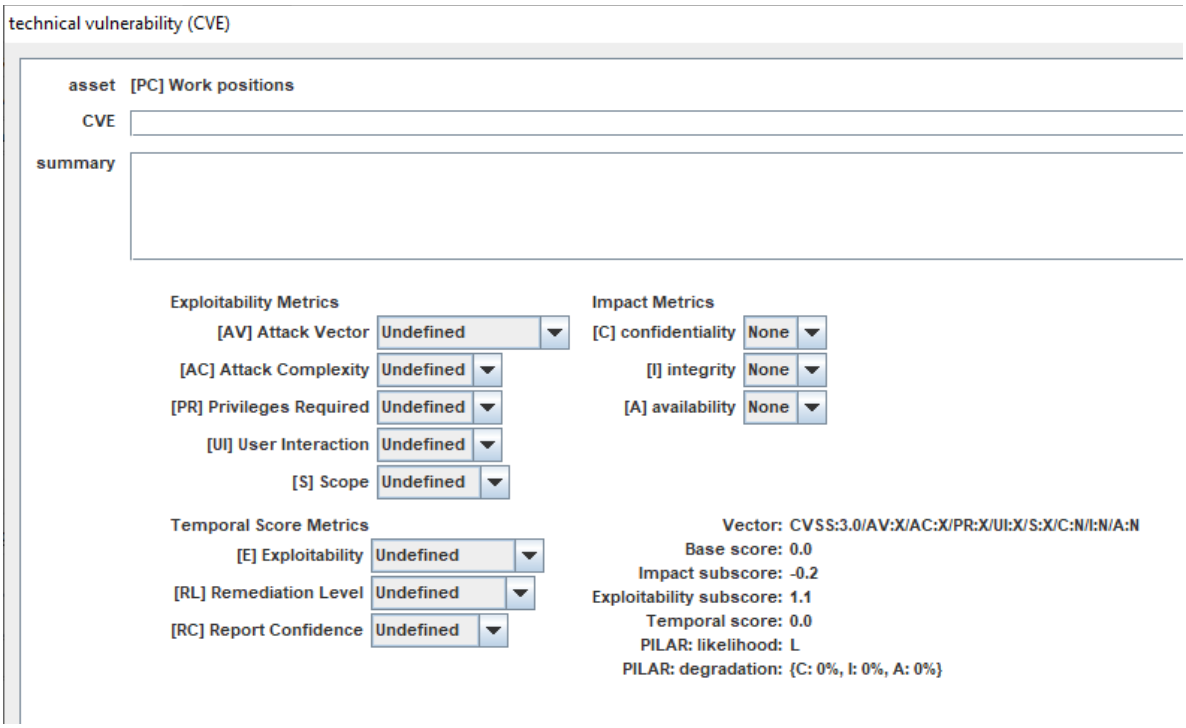
2.1 Manual

Risk analysis >> Threats >> Technical vulnerabilities (CVE)

Select on first column, then click ADD



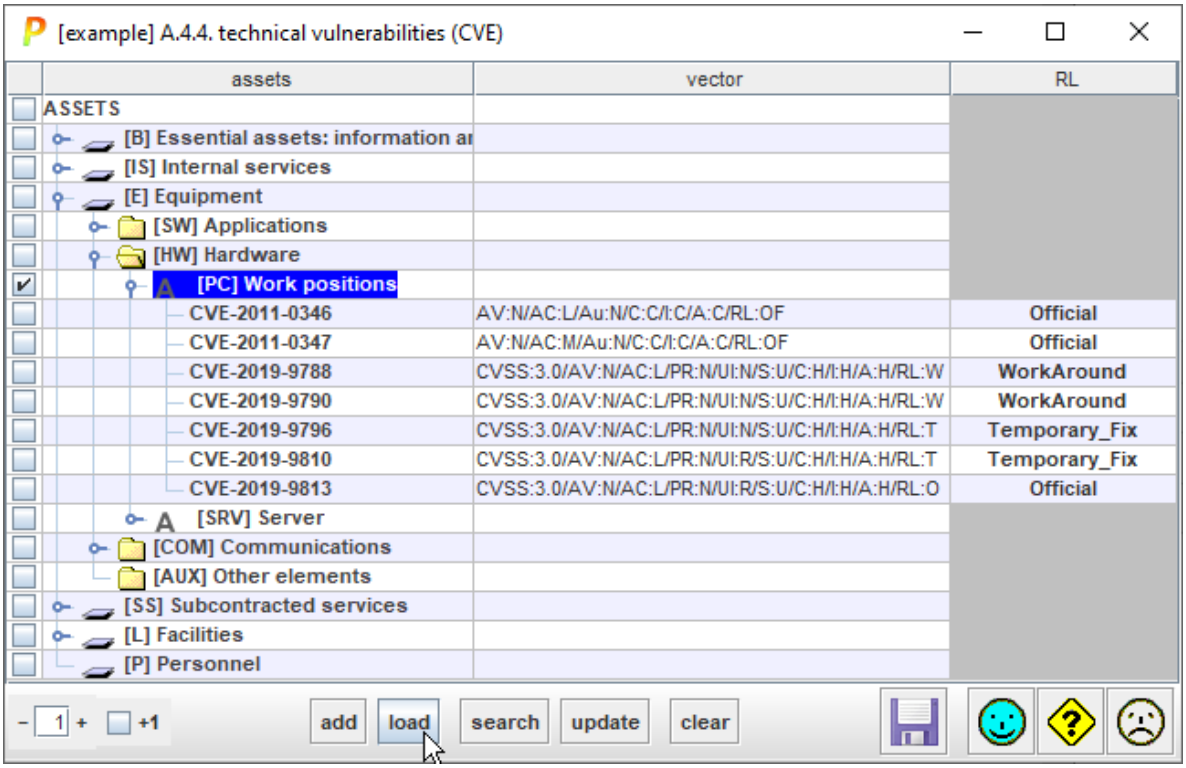
Fill data as appropriate



2.2 From XML

Risk analysis >> Threats >> Technical vulnerabilities (CVE)

Select on first column, then click ADD



See example in section 1.2 above.

These are the minimal fields read by PILAR for CVSS version 2

```
<?xml version='1.0' encoding='UTF-8'?>
<nvd>
  <entry id="CVE-2018-0001 min">
    <vuln:cvss>
      <cvss:base_metrics>
        <cvss:access-vector>NETWORK</cvss:access-vector>
        <cvss:access-complexity>LOW</cvss:access-complexity>
        <cvss:authentication>NONE</cvss:authentication>
        <cvss:confidentiality-impact>PARTIAL</cvss:confidentiality-impact>
        <cvss:integrity-impact>PARTIAL</cvss:integrity-impact>
        <cvss:availability-impact>PARTIAL</cvss:availability-impact>
      </cvss:base_metrics>
    </vuln:cvss>
    <vuln:summary>A remote, unauthenticated attacker ...</vuln:summary>
  </entry>
</nvd>
```

Please, note that XML feeds are no longer supported by NIST National Vulnerability Database.

2.3 From Json

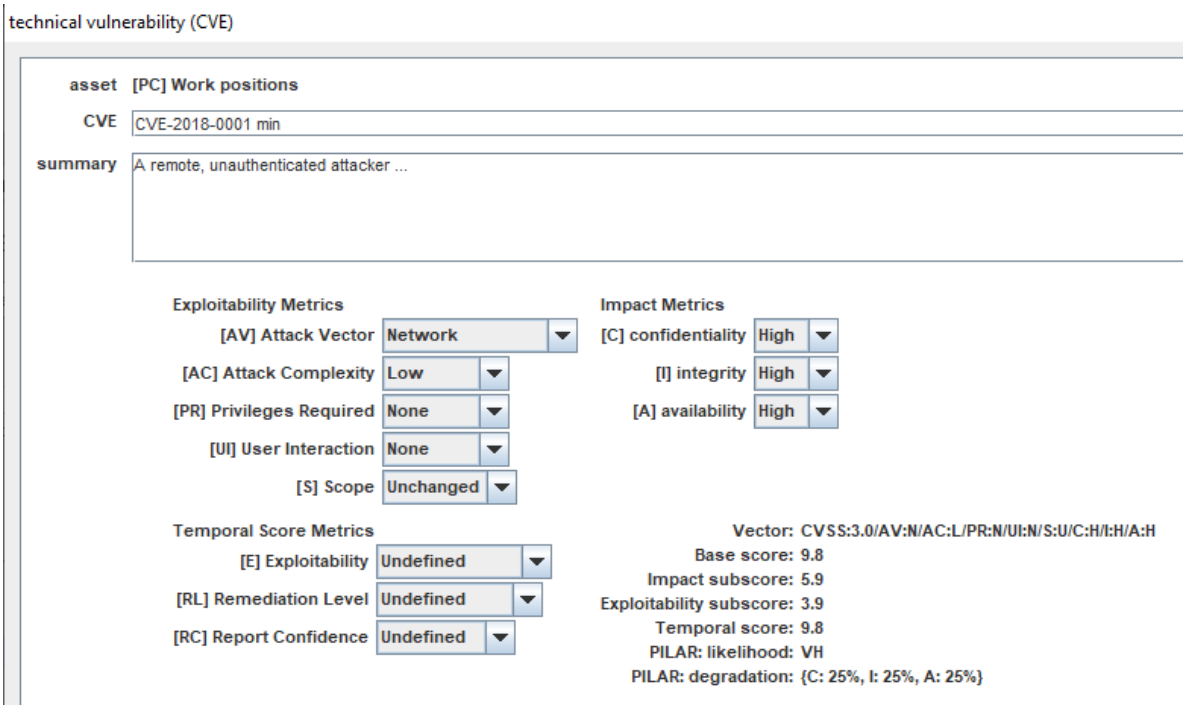
Similar to load from XML.

The minimal fields read from the json files are as follows:

```
{
  "cve" : {
    "CVE_data_meta" : {
      "ID" : "CVE-2018-0001 min"
    },
    "description" : {
      "description_data" : [ {
       "lang" : "en",
       "value" : "A remote, unauthenticated attacker ..."
      } ]
    }
  },
  "impact" : {
    "baseMetricV3" : {
      "cvssV3" : {
       "version" : "3.0",
       "vectorString" :
"CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H",
       "attackVector" : "NETWORK",
       "attackComplexity" : "LOW",
       "privilegesRequired" : "NONE",
```

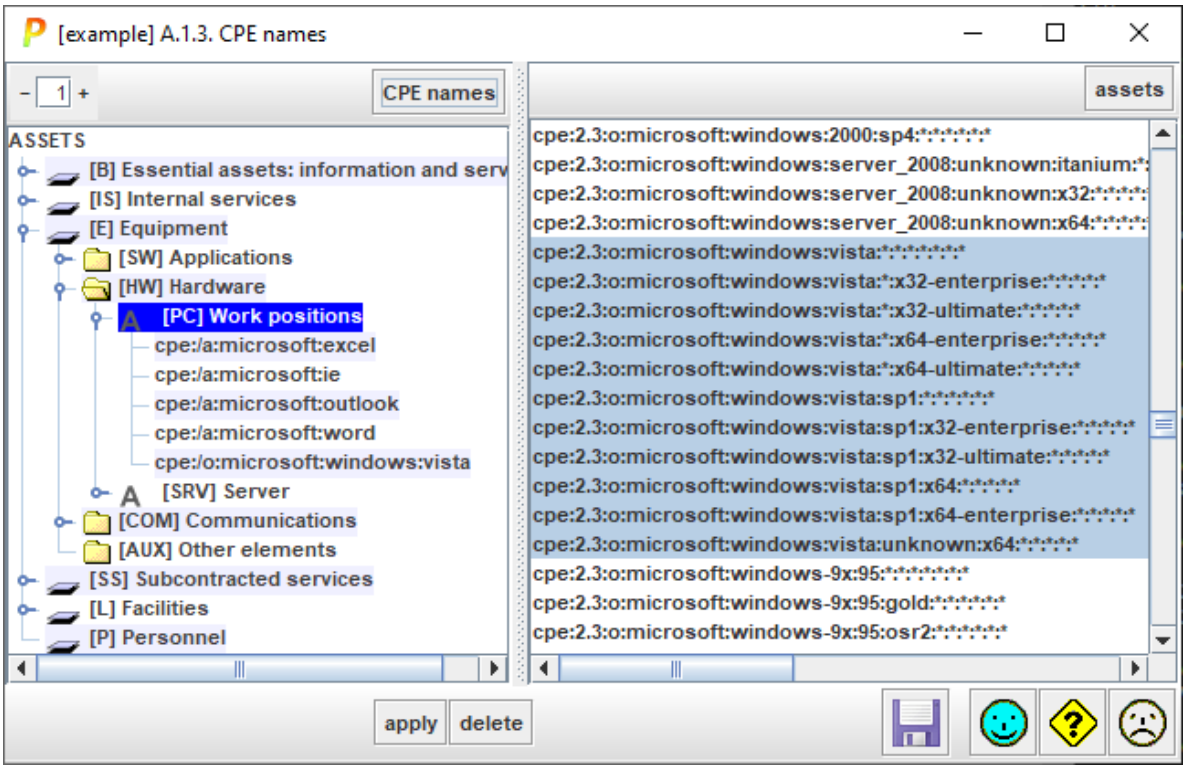
```
          "userInteraction" : "NONE",
          "scope" : "UNCHANGED",
          "confidentialityImpact" : "HIGH",
          "integrityImpact" : "HIGH",
          "availabilityImpact" : "HIGH",
          "baseScore" : 9.8,
          "baseSeverity" : "CRITICAL"
        },
        "exploitabilityScore" : 3.9,
        "impactScore" : 5.9
      },
      "baseMetricV2" : {
        "cvssV2" : {
          "version" : "2.0",
          "vectorString" : "AV:N/AC:L/Au:N/C:P/I:P/A:P",
          "accessVector" : "NETWORK",
          "accessComplexity" : "LOW",
          "authentication" : "NONE",
          "confidentialityImpact" : "PARTIAL",
          "integrityImpact" : "PARTIAL",
          "availabilityImpact" : "PARTIAL",
          "baseScore" : 7.5
        },
        "severity" : "HIGH",
        "exploitabilityScore" : 10.0,
        "impactScore" : 6.4,
        "obtainAllPrivilege" : false,
        "obtainUserPrivilege" : false,
        "obtainOtherPrivilege" : false,
        "userInteractionRequired" : false
      }
    }
  }
```


PILAR reads both CVSS version 2 and version 3, if available. When both are present, CVSS version 3 is preferred.



2.4 Search by CPE

When CPE values are associated to assets



then you may search a feed for CVEs matching the assigned CPE names.

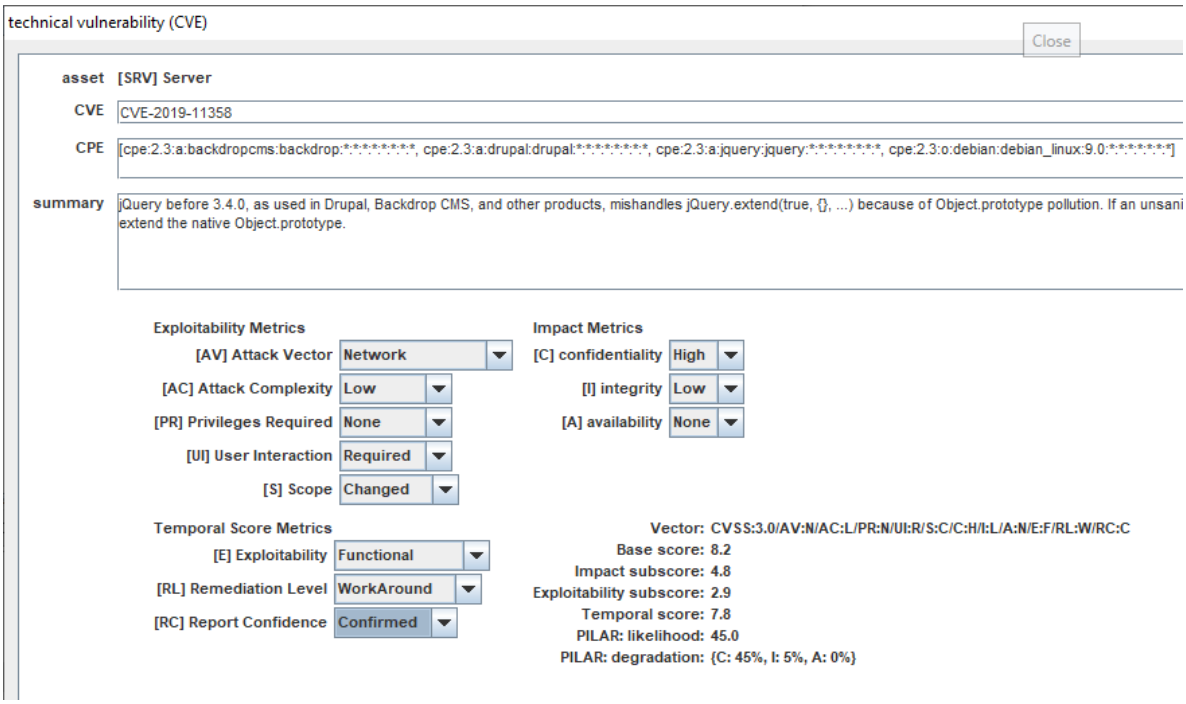
2.5 UPDATE CVEs

UPDATE updates information on the selected CVEs.

3 CVEs in PILAR

CVEs may be associated to assets in PILAR. This association is translated into a security risk. The details are described below.

Example:



CVSS metrics are used to estimate an ARO and a degradation for PILAR:

metric	likelihood	degradation
attack vector	yes	
attack complexity	yes	
privileges required	yes	
user interaction	yes	
scope		
confidentiality impact		yes
integrity impact		yes
availability impact		yes
exploitability	yes	yes
remediation level	yes	yes
report confidence	yes	yes

3.1 Likelihood

PILAR assigns a likelihood to the chances that a CVE is exploited on an asset.

PILAR uses ARO (Annual Rate of Occurrence) as metric. The ARO associated to a CVE is derived from its CVSS metrics.

ARO is estimated as

1 * AV * AC * PR * UI * E * RL * RC

using the following coefficients:

AV	AC	PR	UI	E	RL	RC
X: 1.0	X: 1.0	X: 1.0	X: 1.0	X: 0.5	X: 1.0	X: 0.5
P: 0.5	H: 0.5	N: 10.0	N: 2.0	U: 0.5	O: 0.0	U: 0.5
L: 1.0	L: 0.2	L: 1.0	R: 0.5	P: 0.1	T: 0.1	R: 0.8
A: 2.0		H: 0.33		F: 0.9	W: 0.5	C: 1.0
N: 10.0				H: 1.0	U: 1.0	

Where X stands for 'not defined'. The codes are explained in the next section.

3.2 Degradation

Degradation is the percentage of value that is lost because of the incident. 0% means no loss (tat is, no consequences), 100% means the value is completely lost.

PILAR uses impact metrics to derive degradation,

degradation = impact * E + RL * RC

using an 'order of magnitude' scale:

impact	E	RL	RC
H: 1.0	X: 0.5	X: 1.0	X: 0.5
L: 0.1	U: 0.5	O: 0.0	U: 0.5
N: 0.0	P: 0.1	T: 0.1	R: 0.8
	F: 0.9	W: 0.5	C: 1.0
	H: 1.0	U: 1.0	

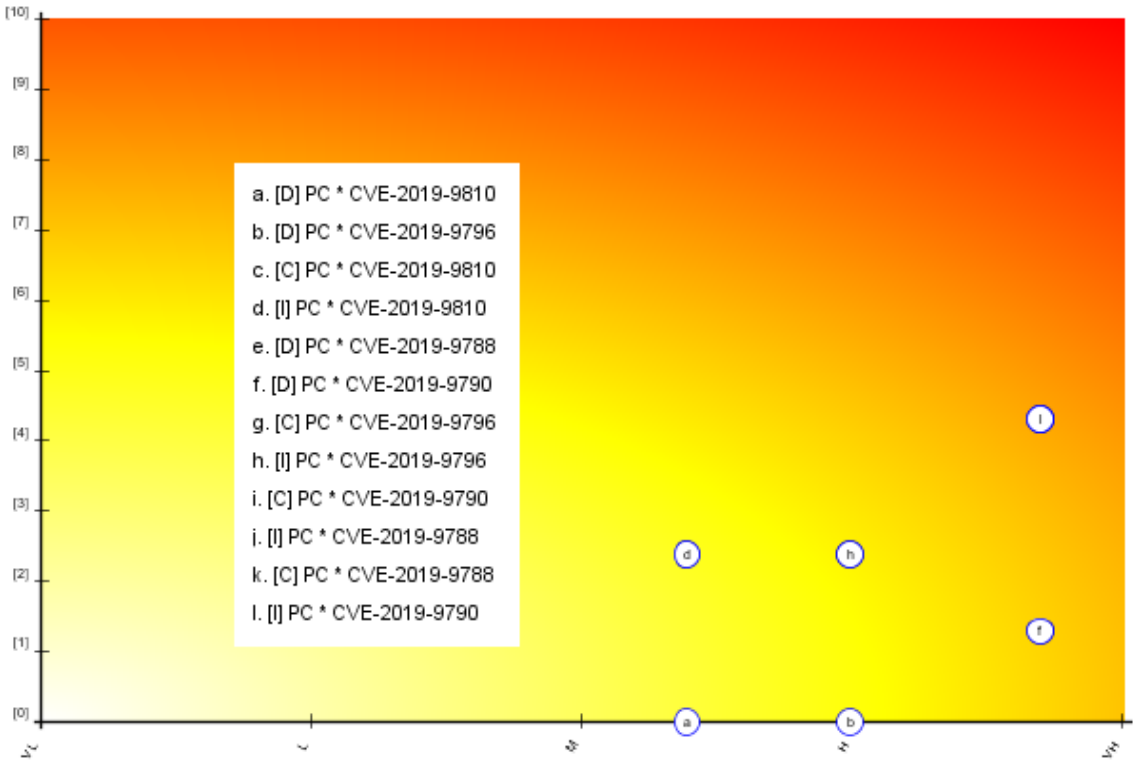
3.3 Risk

Risk is calculated as usual in PILAR

impact = accumulated_value * degradation;

risk = impact * likelihood;

Please, note that for qualitative risk, a heat map table is used, assigning a higher weight to impact and a lower one to likelihood. Refer to Magerit for long reasoning.



4 CVSS fields

Information taken from

Attack vector (AV)

This metric reflects the context by which vulnerability exploitation is possible.

Network (N)	A vulnerability exploitable with network access means the vulnerable component is bound to the network stack and the attacker's path is through OSI layer 3 (the network layer). Such a vulnerability is often termed “remotely exploitable” and can be thought of as an attack being exploitable one or more network hops away (e.g. across layer 3 boundaries from routers).
Adjacent (A)	A vulnerability exploitable with adjacent network access means the vulnerable component is bound to the network stack, however the attack is limited to the same shared physical (e.g. Bluetooth, IEEE 802.11), or logical (e.g. local IP subnet) network, and cannot be performed across an OSI layer 3 boundary (e.g. a router).
Local (L)	A vulnerability exploitable with Local access means that the vulnerable component is not bound to the network stack, and the attacker’s path is via read/write/execute capabilities. In some cases, the attacker may be logged in

Physical (P)	A vulnerability exploitable with Physical access requires the attacker to physically touch or manipulate the vulnerable component. Physical interaction may be brief (e.g. evil maid attack[1]) or persistent.

Attack complexity (AC)

This metric describes the conditions beyond the attacker's control that must exist in order to exploit the vulnerability.

Low (L)	Specialized access conditions or extenuating circumstances do not exist. An attacker can expect repeatable success against the vulnerable component.
High (H)	A successful attack depends on conditions beyond the attacker's control. That is, a successful attack cannot be accomplished at will, but requires the attacker to invest in some measurable amount of effort in preparation or execution against the vulnerable component before a successful attack can be expected.

Privileges required (PR)

This metric describes the level of privileges an attacker must possess *before* successfully exploiting the vulnerability.

None (N)	The attacker is unauthorized prior to attack, and therefore does not require any access to settings or files to carry out an attack.
Low (L)	The attacker is authorized with (i.e. requires) privileges that provide basic user capabilities that could normally affect only settings and files owned by a user. Alternatively, an attacker with Low privileges may have the ability to cause an impact only to non-sensitive resources.
High (H)	The attacker is authorized with (i.e. requires) privileges that provide significant (e.g. administrative) control over the vulnerable component that could affect component-wide settings and files.

User interaction (UI)

This metric captures the requirement for a user, other than the attacker, to participate in the successful compromise of the vulnerable component. This metric determines whether the vulnerability can be exploited solely at the will of the attacker, or whether a separate user (or user-initiated process) must participate in some manner.

None (N)	The vulnerable system can be exploited without interaction from any user.
Required (R)	Successful exploitation of this vulnerability requires a user to take some action before the vulnerability can be exploited.

Scope (S)

the ability for a vulnerability in one software component to impact resources beyond its means, or privileges.

Formally, Scope refers to the collection of privileges defined by a computing authority (e.g. an application, an operating system, or a sandbox environment) when granting access to computing resources (e.g. files, CPU, memory, etc).

Unchanged (U)	An exploited vulnerability can only affect resources managed by the same authority. In this case the vulnerable component and the impacted component are the same.
Changed (C)	An exploited vulnerability can affect resources beyond the authorization privileges intended by the vulnerable component. In this case the vulnerable component and the impacted component are different.

Confidentiality Impact

High (H)	There is total loss of confidentiality, resulting in all resources within the impacted component being divulged to the attacker. Alternatively, access to only some restricted information is obtained, but the disclosed information presents a direct, serious impact.
Low (L)	There is some loss of confidentiality. Access to some restricted information is obtained, but the attacker does not have control over what information is obtained, or the amount or kind of loss is constrained. The information disclosure does not cause a direct, serious loss to the impacted component.
None (N)	There is no loss of confidentiality within the impacted component.

Integrity Impact

High (H)	There is a total loss of integrity, or a complete loss of protection. For example, the attacker is able to modify any/all files protected by the impacted component. Alternatively, only some files can be modified, but malicious modification would present a direct, serious consequence to the impacted component.
Low (L)	Modification of data is possible, but the attacker does not have control over the consequence of a modification, or the amount of modification is constrained. The data modification does not have a direct, serious impact on the impacted component.
None (N)	There is no loss of integrity within the impacted component.

Availability Impact

High (H)	There is total loss of availability, resulting in the attacker being able to fully deny access to resources in the impacted component; this loss is either sustained (while the attacker continues to deliver the attack) or persistent (the condition persists even after the attack has completed). Alternatively, the attacker has the ability to deny some availability, but the loss of availability presents a direct, serious consequence to the impacted component (e.g., the attacker cannot disrupt existing connections, but can prevent new connections; the attacker can repeatedly exploit a vulnerability that, in each instance of a successful attack, leaks a only small amount of memory, but after repeated exploitation causes a service to become completely unavailable).
Low (L)	There is reduced performance or interruptions in resource availability. Even if repeated exploitation of the vulnerability is possible, the attacker does not have the ability to completely deny service to legitimate users. The resources in the impacted component are either partially available all of the time, or fully available only some of the time, but overall there is no direct, serious consequence to the impacted component.
None (N)	There is no impact to availability within the impacted component.

Exploit code maturity (E)

Not Defined (X)	Assigning this value to the metric will not influence the score. It is a signal to a scoring equation to skip this metric.
High (H)	Functional autonomous code exists, or no exploit is required (manual trigger) and details are widely available. Exploit code works in every situation, or is actively being delivered via an autonomous agent (such as a worm or virus). Network-connected systems are likely to encounter scanning or exploitation attempts. Exploit development has reached the level of reliable, widely-available, easy-to-use automated tools.
Functional (F)	Functional exploit code is available. The code works in most situations where the vulnerability exists.
Proof-of-Concept (P)	Proof-of-concept exploit code is available, or an attack demonstration is not practical for most systems. The code or technique is not functional in all situations and may require substantial modification by a skilled attacker.
Unproven (U)	No exploit code is available, or an exploit is theoretical.

Remediation level (RL)

Not Defined (X)	Assigning this value to the metric will not influence the score. It is a signal to a scoring equation to skip this metric.

Unavailable (U)	There is either no solution available or it is impossible to apply.
Workaround (W)	There is an unofficial, non-vendor solution available. In some cases, users of the affected technology will create a patch of their own or provide steps to work around or otherwise mitigate the vulnerability.
Temporary Fix (T)	There is an official but temporary fix available. This includes instances where the vendor issues a temporary hotfix, tool, or workaround.
Official Fix (O)	A complete vendor solution is available. Either the vendor has issued an official patch, or an upgrade is available.

Report confidence (RC)

Not Defined (X)	Assigning this value to the metric will not influence the score. It is a signal to a scoring equation to skip this metric.
Confirmed (C)	Detailed reports exist, or functional reproduction is possible (functional exploits may provide this). Source code is available to independently verify the assertions of the research, or the author or vendor of the affected code has confirmed the presence of the vulnerability.
Reasonable (R)	Significant details are published, but researchers either do not have full confidence in the root cause, or do not have access to source code to fully confirm all of the interactions that may lead to the result. Reasonable confidence exists, however, that the bug is reproducible and at least one impact is able to be verified (proof-of-concept exploits may provide this). An example is a detailed write-up of research into a vulnerability with an explanation (possibly obfuscated or “left as an exercise to the reader”) that gives assurances on how to reproduce the results.
Unknown (U)	There are reports of impacts that indicate a vulnerability is present. The reports indicate that the cause of the vulnerability is unknown, or reports may differ on the cause or impacts of the vulnerability. Reporters are uncertain of the true nature of the vulnerability, and there is little confidence in the validity of the reports or whether a static Base score can be applied given the differences described. An example is a bug report which notes that an intermittent but non-reproducible crash occurs, with evidence of memory corruption suggesting that denial of service, or possible more serious impacts, may result.

5 Glossary

Common Platform Enumeration (CPE)

A nomenclature and dictionary of hardware, operating systems, and applications.

A SCAP specification that provides a standard naming convention for operating systems, hardware, and applications for the purpose of providing consistent, easily parsed names

that can be shared by multiple parties and solutions to refer to the same specific platform type.

https://nvd.nist.gov/products/cpe

Common Vulnerabilities and Exposures (CVE)

A nomenclature and dictionary of security-related software flaws.

An SCAP specification that provides unique, common names for publicly known information system vulnerabilities.

https://cve.mitre.org/

Common Vulnerability Scoring System (CVSS)

A system for measuring the relative severity of software flaw vulnerabilities.

An SCAP specification for communicating the characteristics of vulnerabilities and measuring their relative severity.

https://www.first.org/cvss/

degradation

Degradation measures the loss of value of an asset when a threat occurs.

impact

The magnitude of harm that can be expected to result from the consequences of unauthorized disclosure of information, unauthorized modification of information, unauthorized destruction of information, or loss of information or information system availability. [CNSSI_4009:2010]

likelihood

In Information Assurance risk analysis, a weighted factor based on a subjective analysis of the probability that a given threat is capable of exploiting a given vulnerability. [CNSSI_4009:2010]

risk

A measure of the extent to which an entity is threatened by a potential circumstance or event, and typically a function of: (i) the adverse impacts that would arise if the circumstance or event occurs; and (ii) the likelihood of occurrence.

Framework for Improving Critical Infrastructure Cybersecurity, National Institute of Standards and Technology, February 12, 2014

Security Content Automation Protocol (SCAP)

The Security Content Automation Protocol (SCAP) is a method for using specific standards to enable automated vulnerability management, measurement, and policy compliance evaluation of systems deployed in an organization, including e.g., FISMA compliance. The National Vulnerability Database (NVD) is the U.S. government content repository for SCAP. An example of an implementation of SCAP is OpenSCAP.

vulnerability

In computer security, a vulnerability is a weakness which can be exploited by a threat actor, such as an attacker, to perform unauthorized actions within a computer system. To exploit a vulnerability, an attacker must have at least one applicable tool or technique that can connect to a system weakness. In this frame, vulnerability is also known as the attack surface.