|  |  |  |
| --- | --- | --- |
| logogreen | ASIA-PACIFIC TELECOMMUNITY | **Document:****ASTAP-31/TMP-32** |
| **The 31st APT Standardization Program Forum(ASTAP-31)** |
| 11-15 June 2018, Tokyo, Japan | 14 June 2018 |

Republic of Korea

**Base text: The security guideline for Open Source Software - Rev.1**

This document is agreed as a base text of the security guideline for OSS (Rev.1) in ASTAP-31.

Attachment 1: Base text of the security guideline for OSS (Rev.1)

**The security guideline for Open Source Software - Rev.1**

1. **Introduction**

Open Source Software (OSS) have been used industry, academy, and many other fields these days. However, concern about the security problems about OSS has been issued to introduce OSS in one’s software.

This document describes the issues about using OSS in the software. Part I suggest new vulnerability check model of Open Source Software based on software resource identifier, and Part II informs the guideline to avoid OSS license violence when making a hotfix or patch on one’s product.

1. **Scope of Guideline**

This document is a guideline for who develops and uses software which is built with OSS. The purpose of this document is to provide a comprehensive understanding of security issues with OSS. The scope of the guideline is as follows:

* Vulnerability check model of Open Source Software based on software resource identifying
* Integrated software licensing and vulnerability management guide in the software development lifecycle
1. **Terms and definitions**

OSS, F/OSS:

 Free and Open Source Software

SOFTWARE RESOURCE IDENTIFIER:

 The identifier which indicates the virtual resource file of the software.
*Bill of material* about software.

OSS license:

 Open Source Software license.
Almost every OSS license force that including the copyright notice (includes name of a OSS) in all copies or substantial uses of the work.

**Part I: Vulnerability check model of Open Source Software based on software resource identifying**

1. **Introduction**

Since there is software using Free and Open Source Software (F/OSS or OSS) and the software developers and companies may miss or forget to check the vulnerability in the OSS after when they had finished its development. An OSS used in different software provided by different individuals or companies. Therefore, this model offers to end-users for checking vulnerability on the software by themselves using software resource identifier. When the vulnerability is checked or recognized by the proposed system, it can ensure that the product contains the potentially vulnerable code in software even those function or code is unused or disable, although some of them may not harm or effect when a user is using the product. This proposal can be interpreted as an emergency signal announcement and suggests options to a user before the developer of identified software do any actions for users of their software.

1. **Summary**

This model offers to a system administrator for requesting software resource identifier information from software manufacturers and using a SOFTWARE RESOURCE IDENTIFIER information to check vulnerabilities of their system and components. When the software and its SOFTWARE RESOURCE IDENTIFIER data is given to a system administrator properly, he/she can identify known vulnerability issues which affect their software quickly and precisely.

1. **A virtual model of response model based on SOFTWARE RESOURCE IDENTIFIER metadata**
	1. Overview
		1. Existing response model



In the existing response model, every individual software developers and companies should check vulnerability information about their used OSS by themselves. This kind of information is informed after the information is announced publicly. Even they realized about the vulnerability, no one can force the developers to inform the issues.

Especially, when you compile the software with a package manager, you need to check and identify which version of nested OSS library is used. OSS and modern compiler and package managers now allow a different and broad version of the library for software compilation. Therefore, even you used the same version of OSS library; it does not guarantee that nested OSS libraries are also the same.

Understandably, this model requires source code analysis.

Also, this situation comes from that almost every user is not a direct user of OSS, because the user is insensitive to the OSS vulnerability issues.

* + 1. Response model in this proposal

In this proposal, the vulnerability information is provided by security researcher and companies to a user directly. Even the transmitter does not realize the user’s usage of software; it does not matter.

In this method, a user registers a SOFTWARE RESOURCE IDENTIFIER information about their using software in the receiver.

* + - 1. Summary of flow

Transmitter side: security analyst, researcher

* Vulnerability found
* Vulnerability mapping (Vulnerability DB <-> SOFTWARE RESOURCE IDENTIFIER DB)
* Mapping with public/private vulnerability DB and unique SOFTWARE RESOURCE IDENTIFIER
* Transmitting SOFTWARE RESOURCE IDENTIFIER (& Vulnerability information) signal

Receiver side: Software user (administrator)

* Before receiving
	+ Register SOFTWARE RESOURCE IDENTIFIER of each software
	+ Setup Policies for each situation
* Receiving a signal (SOFTWARE RESOURCE IDENTIFIER)
* Detecting Vulnerable component based on a received signal
* Informing Vulnerability to manager/administrator
* (Responding to vulnerable components like stopping service/blocking it/doing nothing)
	+ - 1. Request software package information

At first, a software user requests the software package information in SOFTWARE RESOURCE IDENTIFIER format. This SOFTWARE RESOURCE IDENTIFIER document contains many metadata that may be a clue to identify the vulnerable component which is going to be checked. This information covers the essential SOFTWARE RESOURCE IDENTIFIER information for checking vulnerability. Using SOFTWARE RESOURCE IDENTIFIER blockchain DB could avoid SOFTWARE RESOURCE IDENTIFIER identifier collision.

* + - 1. Mapping SOFTWARE RESOURCE IDENTIFIER to known vulnerability DB

Once all the necessary vulnerability information and affected SOFTWARE RESOURCE IDENTIFIER information about OSS are present, information mapping can be implemented. Vulnerability database would compare to the SOFTWARE RESOURCE IDENTIFIER element database which contains the unique identifier of the software instead of its source code itself because of efficiency. In the past, the method of checking vulnerability is comparing source code and the vulnerability database or analyzing its properties, and it takes a very long time to complete the process. However, by matching vulnerable OSS source code and its SOFTWARE RESOURCE IDENTIFIER is not delicate task when the vulnerability is found. Mapping can be done with both public or private vulnerability database. Before this process can be done, registration SOFTWARE RESOURCE IDENTIFIER should be done for all software by a developer, and they are responsible for setup policies for every situation.

* + - 1. Emergency alert announces to the software user

Responder(software user/system administrator) receives vulnerability signal and inquires to registered SOFTWARE RESOURCE IDENTIFIER to detect potentially affected software by the received vulnerability.

After checking the vulnerability in the product, a report would be generated automatically. The report shows the vulnerability and potentially affected product and could explain the details of the vulnerability to users. They can read the report in details and take actions like stopping service, blocking it, or no action.

Since checked vulnerability may not be found or neglected by the software developer or vendor, but some security problem would be caused. Therefore, an emergency alert announces to the user and suggest them to shut down or stopped the software to avoid hacker or some cyber-attack before the developer makes an update to fix the vulnerability.

* + - 1. Inform to the software developer

At the same time, vulnerability detail and SOFTWARE RESOURCE IDENTIFIER information would be transmitted to who is responsible for fixing or deal with the vulnerability. The vulnerability information can follow another format if needed, like CYBEX.

1. **Difference between the proposal and existing standards**
	1. CYBEX (X.1500)

This standard is for exchanging cybersecurity information between different stakeholders like researchers, software developers/vendors, and service provider/administrator for security research and fixing. However, in this proposal, vulnerability information is provided by a security researcher/organization to an administrator or a user just for alerting potential vulnerability.

* + 1. SCAP (Security Content Automation Protocol)

SCAP is one of the similar standards with the proposal. Moreover, this covers more than this proposal. However, SCAP and our proposal have a different stance about vulnerability. SCAP finds vulnerability which is confirmed by its developer, but our proposal is standing the problem as an emergency signal.

Our proposal (because vulnerability identifying process is done on user’s computer) once the information mapping process finishes because this does not use string searching or static/dynamic analysis but uses a single value to identify vulnerability, the result is issued at most immediately. SCAP also have some identifying method when the software follows some rules, but cannot be applied to every software. Therefore, users could know security integrity in real time.

More specifically, the process of our proposal can be sure that the product includes vulnerable code in the software, but some of the vulnerable code may not effect when using the product because it is not enabled, reachable nor used.

* 1. ISO/IEC 30111 & ISO/IEC 29147

This two standard provides guidelines on vulnerability handling and disclosure. Based on these two standards, software should be confirmed that there is an attraction of the identified vulnerability. Also, it should try to lower the risk as much as it can. These similar concepts are applied in our model. When vulnerability checked, an alert and temporary suggestion would be provided to a user or infected computer, which can minimize the risk from the vulnerability. Also, the information of the vulnerability would disclose to the developer and user. Besides that, ISO/IEC 30111 also provides guidelines for a vendor to handle the vulnerability to solve the problem. In our model, a report of vulnerability is sent to the developer of the product, and they should take further actions. Before the update published, users are advised to stop using the software.

* 1. ITU-T X.1206

There have been existing similar standard *‘A vendor-neutral framework for automatic notification of security related information and dissemination of updates’* with this proposal. The basic concepts and process of the standard are very similar to this method: Security related information is automatically updated to subscribers. The standard was provided in April 2008. In that time, the OSS software and its development/compilation process were not familiar to every software vendors, and the unique feature of the OSS software couldn’t be reflected the standard.

There are two main differences: 1. Managing software name and version information method, and 2. (Preemptive) potential security information reporting process which shares the same issue with CYBEX/SCAP.

The ITU-T X.1206 identifies the security related information by affected software’s *Version\_Info*, which is consisted of product name, product version, etc. However, the development environment in this day, only the software version could not describe the version information of internal/nested (OSS) libraries. This leads that the user/developer could not realize that the vulnerability is affected in there software (binary).

**Part II: Compliance guide with OSS license when using OSS in your product**

1. **Introduction**

Since there exits software using Free and Open Source Software (F/OSS or OSS) and the software developers and companies may miss or forget to check the vulnerability in the OSS after when they had finished its development. An OSS used in different software provided by different individuals or companies. Because of OSS disclose its source code, the vulnerability is also can be found by others and could be shared publically.

However, when you can fix the vulnerability by yourself, the OSS licenses may force you to disclose your software source code because of how you apply the fix.

In this document, we describe how to fix OSS vulnerability with not disclosing your source code and not violating the licenses.

In these days, software vulnerability should have corresponded immediately. Many developers and organizations involved vulnerability checking and patching process for their software development lifecycle. This kind of approaches is common.

However, when using Open Source Software in their product, they have another aspect to think about their development process.

Open Source Software(OSS) is usually provided with OSS license.

When using OSS for its product, the user/programmer must obligate the clauses of the OSS license.

However, some of the licenses have some strict clauses. For example, viral licenses such as GPL 2.0 or 3.0 requires source code disclosing when you link or include the source code of the software in your software. Some of the licenses such as LGPL family allows to just linking to your software without source code disclose, but when you include the source code on your software directly, then you may need to disclose your source code depending on how and when you apply the hotfix to your software.

Unfortunately, we cannot give credible and reliable advice about OSS license interpretation because the meaning of many OSS licenses are not clear there are not so much ruled cases about OSS license involved cases. Also, many cases depend on de jure, especially on the clauses of copyright, etc. therefore following guides may not suitable for applying your region and cases.

However, applying a patch to fix a vulnerability should be done immediately. When a developer or company have abundant resources to deal with the problem, this may not be the problem.

1. **Avoid using OSS which provided under viral licenses or introducing OSS component tracking system into the software development process**

Avoid using OSS which provided under viral licenses security-critical software when you want to keep the source code of software closed.

The range of source code disclose caused by source code editing can be various. This range depends on when the source code is edited, how the OSS is linked to their software, and many other factors based on the OSS licenses.

If you want to keep the software closed, the software development process should be designed by included OSS with their licenses. Design process to manage licenses, the development process could monitor where the hotfix is applied and which components links with the applied component. Many of present software development process still not monitor this kind of forms.

When you cannot improve your software development process, you should exclude OSS from your choices for your products.

If the product needs immediate corresponding to a security problem, this kind of monitoring should be done automatically. Checking the linking form, its relationships, and its license interpretation

1. **OSS isolating design and reordering hotfix patching step**

When you cannot avoid using viral licensed OSS, 1) design your software to isolate from OSS component and your software carefully, or 2) Make sure patching step is not violated disclosure clauses.

1. When the viral licensed OSS component is isolated from your software, many cases do not violate the licenses.

2. However, when you cannot isolate OSS component from your software (such as embedded software, etc.), you make sure that the order of applying the hotfix.

When you follow the proper step to apply the hotfix, you may not lead license violation without disclosing the whole of software.

1. **Check the nested library’s detailed version**

When you compile the software with a package manager, you need to check and identify which version and license of nested OSS library are used. OSS and modern compiler and package managers now allow a different and broad version of the library for software compilation. Therefore, even you indicate specific OSS library by its name; they may consist of different version of OSS library, which may be changed its license.

**References**

**(WIP)**

* CYBEX / SCAP
* ITU-T X.1206
* ISO/IEC 30111
* ISO/IEC 29147