Introduction to the OWASP Top 10 – 2021

Risk A6: Vulnerable and Outdated Components

Key Concepts

Key Concept 1: Previous Versions of Software Have Vulnerabilities

- These vulnerabilities are fixed in new versions of the software

Key Concept 2: End of Life and End of Sale

- End of Life (EoL): the date software support stops
  - The developer community or company stops supporting the software on this date
  - Includes ending security updates for software and hardware
- End of Sale (EoS): the last day to order a product through a vendor
  - For a period after the EoS date is announced, a vendor may provide continued support for hardware and software issues
- Unsupported Software: the operating system of a 10-year-old computer that is no longer supported

Key Concept 3: Attackers Attempt to Find and Exploit Vulnerabilities

- There are teams investigating and analyzing available information to identify potential vulnerabilities
- Out-of-date software is on top of their list

Additional Information

Previous versions of software more than often have vulnerabilities within them. These vulnerabilities are fixed in new versions of the software.

Systems and frameworks have important dates regarding their support. End of Life (EoL) is the date by which software support stops. The developer community or company stops supporting the software on this date. This also includes stopping providing security updates for software and hardware.

End of Sale (EoS) is the last day by which you can order a product through a vendor. For a period after the EoS date is announced, a vendor may provide continued support for hardware and software issues.

Attackers attempt to find and exploit vulnerabilities in software and hardware. There are teams out there with the sole purpose of investigating and analyzing available information. They do this to identify potential vulnerabilities and out of date software and hardware is on top of their list.

Definition

Having, in use, software components that are vulnerable, unsupported, or out of date.

- Specific vulnerabilities are known, but unable to be patched in time
- Patching is not available for your systems based on specific software dependencies
- Think of patching as a quarterly, or monthly process. Then think of the latest vulnerability for a specific library. These two frequencies are often different i.e., a vulnerability is released at a time that does not coincide with your patching schedule.
- Your organization does not know what software components and versions in use

If you do not scan for vulnerabilities regularly, or if you do not subscribe to security bulletins related to the software components, you use, this can be a bigger problem.

Additional Information

The definition of A6 of 2021 – Vulnerable and Outdated Components – is having in use within your systems and applications, software components that are vulnerable, unsupported, or out of date.

This could be because you know of the specific vulnerabilities but are unable to patch them in a timely manner. It could also be because patching is not available for your systems based on specific software dependencies. Think of patching as a quarterly, monthly, or weekly process. Then think of the latest vulnerability for a specific library. Commonly, these two frequencies are different i.e., a vulnerability gets released at a time that does not coincide with your patching timings.

Example

The example below references the Apache Log4J library, an open-source logging framework which is very popular for Java web applications.

Bad Example

```
<dependency>
  <groupId>org.apache.logging.log4j</groupId>
  <artifactId>log4j-core</artifactId>
  <version>2.14.1</version>
</dependency>
```

Good Example

```
<dependency>
  <groupId>org.apache.logging.log4j</groupId>
  <artifactId>log4j-core</artifactId>
  <version>2.16.0</version> <!-- or newer -->
</dependency>
```

Additional Information

In December 2021, log4j was found to be vulnerable to remote code execution.

As a bad example, of what not to do, we see the dependency on a version of log4j that contains this vulnerability.

As a good example, we are using a later version of log4j that has been patched against log4shell, which is the specific exploit in question.

Challenges

- Not a regular process that coincides with when a specific vulnerability is discovered
  - Patching software for vulnerabilities might be a regular process within organizations, but is not a regular process that necessarily coincides with when a specific vulnerability is discovered
  - For patching to work, you must know what software versions your organization is using
  - It can be complex process to get right in terms of timing
- The process for ensuring 3rd party libraries are updated is complex and time-consuming
  - Your software uses a vast number of libraries and has typically dependencies in the 100s if not the 1000s
- Updating 3rd party libraries is neglected

Additional Information

We now understand how the risk of vulnerable and outdated components can materialize, let us look at why it is a common issue.

Patching software for vulnerabilities might be a regular process within organizations but is not a process that necessarily coincides with when a specific vulnerability is discovered. For patching to work, you must know what software versions your organization is using.

Your software uses a vast number of libraries and has typically dependencies in the 100s if not the 1000s. Thus, the process of ensuring 3rd party libraries are up to date can be complex and very time-consuming. Finally, developers have other tasks to do, so at times updating 3rd party libraries is neglected.

Best Protection Strategies

CORUS

Check periodically your libraries are updated

Only obtain components from official trusted sources

Remove unused dependencies

Use only features that are necessary

Stay current with latest vulnerabilities

Additional Information

Above you have some of the best protection strategies against A6 of the Top 10 – Vulnerable and Outdated Components.