



Unit 2: UML Modelling to Support Secure System Planning

LEARNING OUTCOMES:

- **On completion of this unit you will be able to:**
 - Research the academic literature on software development which follows an agile process and prioritises security.
 - Break down the steps involved in a process so that they may be represented as a flow chart.
 - Reading areas of an ISO/IEC Standard on security and becoming familiar with the language and concepts.

NOTES FROM: DADZIE, J. (2005) UNDERSTANDING SOFTWARE PATCHING, *IEEE*.

- Even under ideal conditions, however, problems always arise. Most software will be used for many years in an ever-changing user environment. This can place new compatibility demands on software and introduce new security vulnerabilities not originally envisioned. Whatever their source, problems can be found in any piece of software and must be addressed with **patches**.
- **Definition(s):** The systematic notification, identification, deployment, installation, and verification of operating system and application software code revisions. These revisions are known as patches, hot fixes, and service packs.

Source(s):

CNSSI 4009-2015

NIST SP 800-137 under Patch Management from CNSSI 4009

THE SOFTWARE PATCHING LIFECYCLE


- IDENTIFYING THE PROBLEM
- Security-related patches are common in the software development world. In many cases, security researchers and hackers find vulnerabilities missed during the development cycle, but software vendors find some themselves after the product ships. In the best case, those who find a problem will notify the vendor immediately, before publicly announcing the vulnerability. Other times they do not, however. In some cases they even post exploit code publicly prior to availability of a fix, thereby greatly increasing the risk to users of the affected component. Regardless of the source of the vulnerability, the software vendor has a responsibility to research the issue and, if valid, produce a patch to address the problem and distribute it as widely as possible.
- Developers should not consider the patch development process complete until threat models, design specs, test plans, and code analysis tools are updated to ensure that similar types of vulnerabilities are caught when new software is being developed

PATCH MANAGEMENT

- No matter how clearly described, patch packages can't be easily deployed unless they can be easily installed. This means that packages must support **silent installation** through the command line of the target operating system. The silent installation options must be the same for all patches for the same application to ease the administrative cost of incorporating the patch into patch management solutions.
- Therefore, it's important for **a patch installation to log its activity to a common log** so the patch application process can be tracked for reporting and debugging of problems if they occur. Patches should also support uninstall or rollback. This is especially important if problems occur with the patch. The uninstall or rollback process must account for patches that may have been installed out of order with respect to when they were released.
- Finally, patches must be protected from tampering and their integrity verifiable through **digital signatures**, hashes, or checksums. Digital signatures are preferred since the source of the patch can be also be verified

PATCH MANAGEMENT

- Patch management is the process of distributing and applying updates to software. These patches are often necessary to correct errors (also referred to as “vulnerabilities” or “bugs”) in the software.
- Common areas that will need patches include operating systems, applications, and embedded systems (like network equipment). When a vulnerability is found after the release of a piece of software, a patch can be used to fix it. Doing so helps ensure that assets in your environment are not susceptible to exploitation.
- Patch management is important for the following key reasons:
 - **Security:** Patch management fixes vulnerabilities on your software and applications that are susceptible to cyber-attacks, helping your organization reduce its security risk.
 - **System uptime:** Patch management ensures your software and applications are kept up-to-date and run smoothly, supporting system uptime.
 - **Compliance:** With the continued rise in cyber-attacks, organizations are often required by regulatory bodies to maintain a certain level of compliance. Patch management is a necessary piece of adhering to compliance standards.
 - **Feature improvements:** Patch management can go beyond software bug fixes to also include feature/functionality updates. Patches can be critical to ensuring that you have the latest and greatest that a product has to offer.
 - Source: <https://www.rapid7.com/fundamentals/patch-management/>

- 
- Although the agile development approach is getting acceptance across the globe, it is found to have certain disadvantages related to security in software development [2, 3].
 - The main security issues with agile development arise from the informal communication, self-organizing team, tacit knowledge-driven methods and trust on individuals, as they conflict with the assurance and quality activities as required by conventional secure software development methods.
 - Some limitations are imposed on the projects by the agile processes, as all the requirements are not known in advance.

SCRUM

- TO CHECK THIS SITE
- <https://www.pmi.org/learning/library/agile-project-management-scrum-6269>