Software Upgrades and Updates

- Importance of software updates
- Avoiding malicious software updates
- Sources for software updates
- Software updates deployment and release procedures
- Risk assessment and impact analysis for updates
- Backup and rollback procedures in case of update failures

Importance of software updates

Software needs to be regularly updated for various reasons including responding to new technology changes, preventing security issues, improving compatibility and enhancing program features. Some of the key reasons for making software updates include:

i. Patch security flaws

Changes to manage security flaws should be made as soon as they are available to address software vulnerabilities that enable cybercriminals to gain unauthorized access to an institution's computing resources.. Threat actors see these vulnerabilities as open doors, enabling them to plant malware for illegal purposes such as data theft, sabotage, fraud or for blackmail purposes.

Malware enables threat actors to take control of computers and steal information. Malware can also encrypt files, documents and other programs so they are unusable. Security patches block these open doors in the software to protect a device from attacks.

It is therefore important for Government institutions to keep software updated by deploying patches and updates provided by the software vendors as these fix vulnerabilities that are identified in the software. Any patches deployed should only be sourced from official sources provided by the vendor.

ii. Get new features

Installing updates may add new features and remove old ones that are no longer necessary. Technology is constantly changing and updates offer the latest features and improvements.

iii. Improve performance

Software vendors may find bugs in a program or need to make necessary enhancements to a program to improve the performance of the software.
Electronic devices also need regular maintenance and routine updates to run their best. Having the latest patches can help prevent software from crashing.

iv. Ensure compatibility

Software manufacturers send updates to ensure their software is compatible with the latest technology. Without updates, older software may not be able to work with newer technology.

Avoiding malicious software updates

Government software maintenance teams and users should also be informed to be aware of fake update messages. These messages are another way threat actors try to access computers and mobile devices.
Malicious messages can appear to users in the following ways:

- **Pop-ups.** They may look real but clicking on a pop-up may lead to a hack. Users should be advised to avoid malicious pop-ups. Updates should only be obtained from verifiable sources such as directly from the vendor website. Updates should also appear on the software's website, app store or within the program.
- **Emails.** The email may state that an update is long overdue. It may even have the company logo with all the correct spelling in the text. Don't click on links or respond to these types of emails. For information about updates, it's best to contact the software's support desk or check the vendor website.

One of the best ways to avoid malware downloads is to let the software automatically update where available. Many external vendor software solutions have an option to automatically run updates when available.
Mobile devices are increasingly being used for work purposes. Users should therefore also be reminded to keep their mobile devices updated. They should check for software update notices in the settings of their mobile devices.

Sources for software updates

A software update is a change or fix to a program that replaces the older version. There may just be a few tweaks to improve the way it works, or it could offer entirely new features. Updates or patches are also done to fix security vulnerabilities. These updates are usually provided at regular intervals by the software manufacturer and are usually downloaded from the internet to the device. Software updates can also come from internal development teams.

It is important to ensure that updates or patches are only obtained from a genuine source provided by the software manufacturer to avoid installing malicious software. Official sources usually include websites provided by the software manufacturer or automatic updates. Government Institutions should follow instructions from the software manufacturer on how to get the updates and how to properly install them.

Software updates deployment and release procedures

Deployment is a shift of software from one controlled environment to another. On the other hand, releases are a collection of changes delivered to the Production environment for users to experience. Applications need several updates, security patches, and code changes. Deploying them across platforms and environments requires proper management of releases.

The following process should be followed to deploy software changes:

Preparation

During this stage, developers test if the code is running smoothly and ready to deploy. Then, they must gather all the code along with all the configuration files, libraries, and resources needed for the seamless functioning of the application. These items combined create a single package to be released. Important activities to carry out during preparation stage:

- Notify all the stakeholders about the upcoming deployment.
- Identify and inform all the collaborators to reduce friction between the teams involved in the software development lifecycle (SDLC) process.
- Check third-party software requirements to ensure everyone knows their uses to minimize any issues in the future.
- Map out the deployment process to ensure everyone is on the same page.
- Have a rollback plan ready in case any critical issues occur during deployment.
- Establish key performance indicators (KPIs) to measure the deployment's effectiveness and determine whether it succeeds or fails.

Testing

The objective of the testing stage is to validate the software update or change before it reaches the deployment stage. During this phase, the update is deployed to a test environment and subjected to testing. Developers then review the results of the testing to fix any errors before deploying the update to a live environment. Key activities carried out include:

- Conducting unit tests on components of the software affected by the change to verify that the results meet user requirements.
- Integrating unit tests with the continuous integration (CI) tools where available to automatically build and verify each portion and remove bugs easily.
- Creating a staging environment similar to target the production environment and run tests to ensure the code works as intended.
- Run end-to-end tests to check the application's workflow and its compatibility with other components like hardware and network connectivity.
- Seek feedback from stakeholders including requesting end users to undertake functional testing to determine whether the code is ready for production.

- Use smoke tests after deployment to make sure the software is running smoothly.

Deployment

Once the update is thoroughly tested, you can deploy it to the live environment. At this stage, the main task is to look out for any bugs and errors that might occur on the live server and fix those to deliver the best experience possible. Key activities carried out include:

- Deploy to the production environment to let users interact with the software.
- Monitor performance closely by matching it with the predetermined KPIs and checking for errors.
- Automate rollbacks so you can go to the previous version if the release is unsuccessful.
- Check environment health to identify potential issues with the operating system (OS) or database.
- Keep track of logs to gain insight into how the software is running and identify any security threats.
- Document everything to maintain consistency when you want to change the new version.

The software deployment process can be manual or automated, but it should be done at the least disruptive time in an organization’s work routine.

DevOps tools such as continuous delivery software can enable developers to generate deployment-ready code quickly. This means software releases remain in a deployable state, enabling automatic deployment to production within seconds.

Risk assessment and impact analysis for updates

Risk is the future uncertainty between potential events and the potential for loss. Classic software documentation states that every deployment process should undergo a thorough inspection. This may increase test costs but reduce risks of failure. Some guidelines to mitigate risks of software guidelines include:

- Use legitimate sources of updates. Only download software updates from trusted sources to avoid introducing malicious software
- Implement continuous integration and delivery. Continuous integration and delivery (CI/CD) is a set of practices and tools that automate the building, testing, and deploying of a system. CI/CD enables you to deliver frequent and reliable updates to the system, as well as to detect and fix errors quickly. CI/CD also reduces the risk of conflicts and inconsistencies among the system components, as well as the need for manual interventions. Some common CI/CD tools are Jenkins, Travis CI, and CircleCI.
- Test thoroughly. Testing is a crucial step in ensuring the quality and functionality of a system. Testing should be done at different levels, such as unit testing, integration testing, system testing, and user acceptance testing. The system should also be tested in different environments, such as development, staging, and production. Testing will help you verify the system requirements, identify and resolve defects, and evaluate the system performance and usability. You should also use automated testing tools and frameworks, such as Selenium, JUnit, and TestNG, to speed up and simplify the testing process.
- Monitor and evaluate. Once you deploy the system or make changes to it, you should monitor and evaluate its performance and behavior. You should use tools and metrics that can track and measure the system availability, reliability, scalability, security, and user satisfaction. You should also collect and analyze feedback from the stakeholders, users, and customers, as well as from the logs and reports generated by the system. Monitoring and evaluating will help you identify and troubleshoot any issues or anomalies, as well as to improve and optimize the system over time.
- Consider taking a backup of the software before deployment
- Changes should be deployed outside of office hours and users should be informed of the changes

Backup and rollback procedures in case of update failures

System failures and errors can happen at any stage of a system implementation, from development to testing to deployment. To mitigate this risk the following should be put in place:

- Before starting implementing of a new system or making changes to an existing one, a clear plan of action should be put in place. This includes defining the scope, objectives, requirements, risks, and dependencies of the project, as well as the roles and responsibilities of the team members.
- Define a backup and restore strategy, a rollback and recovery plan, and a communication plan in case of emergencies. Having a plan will helps to avoid or mitigate potential issues and ensure a consistent and quality outcome.
- Use version control - version control is a system that tracks and manages the changes made to the source code, configuration files, and other artifacts of a system. It allows one to create, review, and merge different versions of the system, as well as to revert or roll back to a previous version if something goes wrong. Version control also facilitates collaboration and coordination among the team members, as well as documentation and auditing of the system changes. Some popular version control tools are Git, Subversion, and Mercurial.