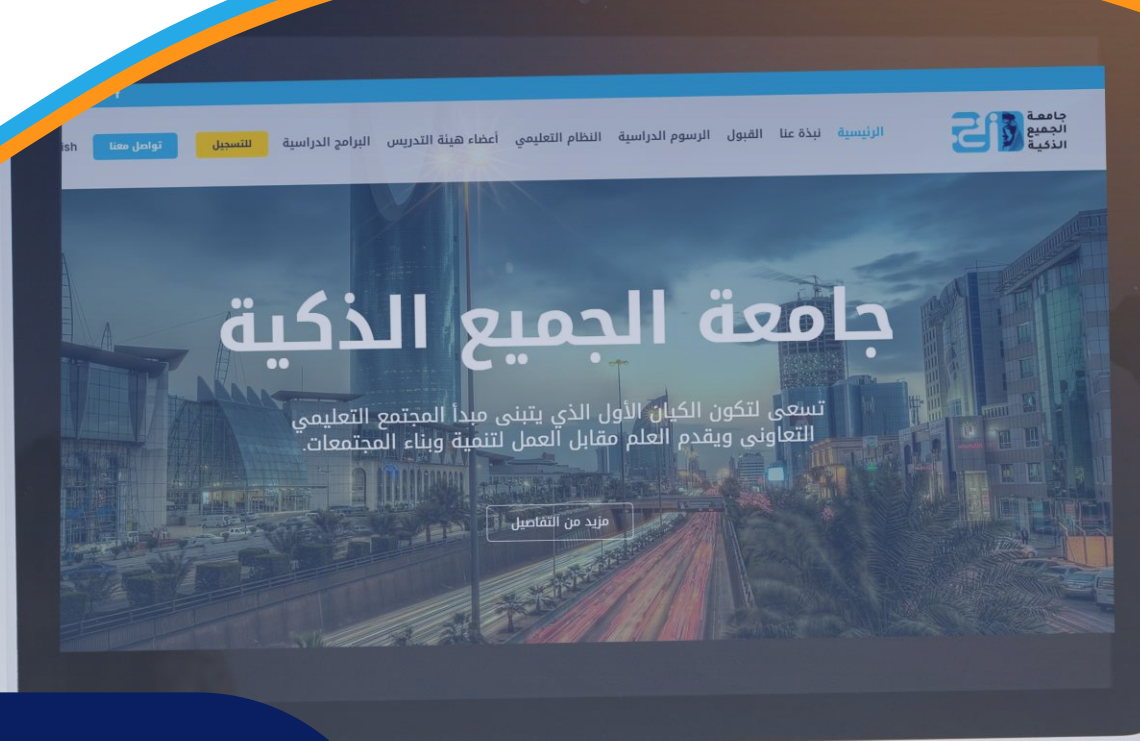




جامعة الجميع الذكية
EVERYONE'S SMART UNIVERSITY



IoT Security
Prepared by Dr. Abdelrhman Ahmed
Lecture 10

Manage
Software & Firmware Risks

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.

Our software and applications need **patches in order to fix security flaws, bugs and add feature developments**. However, in this case, patching is not a temporary task, but a **proactive planned technique**.



Patch management is more than just updating and repairing IT software. It remediates vulnerabilities and manages risk. Patching is a sub-category of risk-based prioritization and software lifecycle management. Once you detect a critical vulnerability in the applications or operating system, you need to seek a resolution. It may involve removing an old certificate, changing a configuration or updating software with a patch.

Applying patches to hardware and software applications may bring about additional risks to the IT environment because patches are programs and may have their own set of vulnerabilities. If a patching process is not handled properly it may lead to system crashes or damage hardware devices.



Invalidated Patches

The source of each patch must be validated by scrutinizing the acquiring source and patch signature to determine only authenticated patches are applied to organization information systems. It has been reported that some cybercriminals sent fake Microsoft security patch emails with malicious content. Additionally, some complex patches need domain expertise to examine the certain prerequisites and dependency metadata before the installation. Failure to do so may cause severe consequences such as unpredictable system behaviors, data corruption or service outage.

Insufficient Testing

Many organizations' information systems are connected and have interfaces among them to exchange data. Applying patches to a system in the production environment without adequate testing may bring negative impact to other applications such as incompatible communication protocol, data formats or interface logic.

Vulnerabilities in Patch Management System

If patch management is used to implement an automated patching tool, the security vulnerabilities might have an impact on other information systems. A breached or virus-infected patch management system will be a central distribution point that transmits malware and viruses. Additionally, a patch management system protected with weak access controls creates an additional channel for cyber attackers to gain unauthorized access to the organization's IT environment or launch attacks on the essential information systems.

Downtime and Interruption

With the increase in complexity of the program, patches are released regularly and need longer time spent on installation on the target information systems. Patching tasks, if not done professionally, may lead to frequent interruption to organizations' operations and prolonged service downtime because of large patches.



Incorrect Identification and Installation

Detection and deployment of security patches is a major feature of the patch management and software patching process. Some sophisticated applications have functions embedded to detect application security patches and offer needed guidelines on the patch installation processes. Making use of alternative means to identify and install patches is risky because the reliability and accuracy will not be guaranteed by the vendors.

Inadequate Fallback Procedures

Sometimes the vendor may publish a patch that has errors and leads to different issues related to patch systems. If organizations do not have the right fallback processes, the adverse effect imposed by that patch issue can't be quickly reversed until the vendor provides another patch to fix the mistake.

Incorrect Identification and Installation

Detection and deployment of security patches is a major feature of the patch management and software patching process. Some sophisticated applications have functions embedded to detect application security patches and offer needed guidelines on the patch installation processes. Making use of alternative means to identify and install patches is risky because the reliability and accuracy will not be guaranteed by the vendors.

Inadequate Fallback Procedures

Sometimes the vendor may publish a patch that has errors and leads to different issues related to patch systems. If organizations do not have the right fallback processes, the adverse effect imposed by that patch issue can't be quickly reversed until the vendor provides another patch to fix the mistake.

Chapter 10

Manage Risks related to Patch Management

- **Develop an up-to-date inventory of all your production systems:** Whether this be on a quarterly or monthly basis, this is the only way to truly monitor what assets exist in your ecosystem. Through diligent asset management, you'll have an informed view of operating systems, version types, and IP addresses that exist, along with their geographic locations and organizational "owners." As a general rule, the more frequently you maintain your asset inventory, the more informed you're going to be.
- **Devise a plan for standardizing systems and operating systems to the same version type:** Although difficult to execute on, standardizing your asset inventory makes patching faster and more efficient. You'll want to standardize your assets down to a manageable number so that you can accelerate your remediation process as new patches are released. This will help save both you and technical teams time spent remediating.

Chapter 10

Manage Risks related to Patch Management

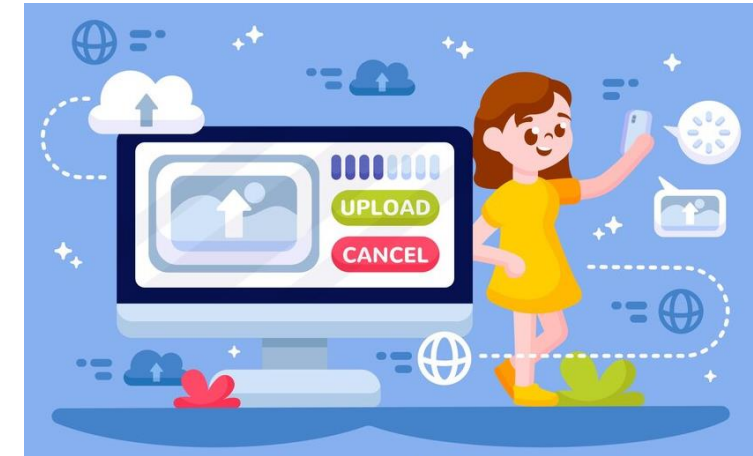
- **Make a list of all security controls that are in place within your organization:** Keep track of your firewalls, antivirus, and vulnerability management tool. You'll want to know where these are sitting, what they're protecting, and which assets are associated with them.
- **Compare reported vulnerabilities against your inventory:** Using your vulnerability management tool to assess which vulnerabilities exist for which assets in your ecosystem is going to help you understand your security risk as an organization.
- **Classify the risk:** Through vulnerability management tools you can easily manage which assets you consider to be critical to your organization and, therefore, prioritize what needs to be remediated accordingly.

Chapter 10

IoT OTA Updates

Updates are critical for all technology, but especially IoT devices because they're often located in the field, on the factory floor or healthcare center. Organizations use connected devices to track inventory, manage machines, increase efficiency, save costs and save lives. Making the update process easy and secure is critical to business. **Over-the-air (OTA) updates** offer organizations these advantages for IoT devices. OTA updates use the same wireless communication that the devices already send and receive data through.

However, OTA updates of IoT devices are not the silver bullet organizations want. IoT OTA updates pose a set of six challenges and risks that **could cost organizations time, effort, money, and resources.**



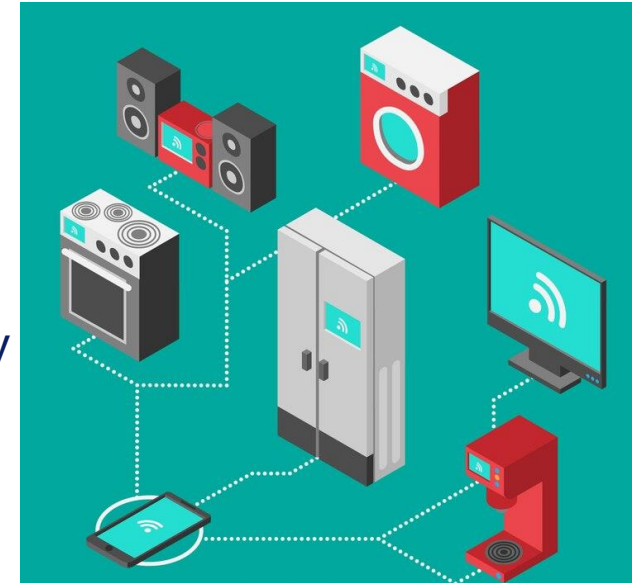
Create context-aware updates

Many organizations use an IoT platform to manage updates for their hundreds or thousands of IoT devices. In today's 24/7 work environment, however, even the best platforms struggle to update all IoT devices.

When IoT devices run every moment of the day, updates will reach the devices when they're in use. That could force the device to stop its primary function to handle the update and disrupt the entire work process. Alternatively, the device will prioritize its work and ignore the update entirely until its ready, which might be days, weeks or even months in the future. Depending on the update, this could lead to future device disruption, increased cybersecurity risk or device failure. **Ideally, updates should be sent to IoT devices when they're free from their regular tasks to avoid disrupting work processes.** Administrators could also schedule partial updates to save time and processing power. For example, by sending a partial firmware update, the device could apply only the necessary changes to the onboard firmware image, reducing the total update time.

Plan for connectivity issues

A device's hardware and software may make it difficult to receive and perform an OTA update because it cannot establish and maintain a network connection long enough. Maintaining a connection for the duration of an update is especially important for devices that are not normally continuously connected to the network. Devices with hardware connectivity issues are easy to detect and fix, whereas software connectivity issues are not. Bad or failed software updates due to a connectivity issue can brick a device or fleet. **To avoid bricking devices, perform a connectivity test before sending updates to devices.** Remove any updates that are unable to sustain the connection from the update list and schedule them for maintenance.



Prepare for recovery from failed or interrupted updates

OTA updates should be an all-or-nothing proposition because they either succeed or fail completely. When they fail, they should do it in a way that the device can recover functionality and not disrupt work processes and alert IT staff of the issue. For example, an IT administrator could set devices to automatically roll back an update if network connectivity isn't restored in a set time or enable always-on network connectivity during the entire update process for devices that don't remain in constant contact with the network.



Reduce security risks

Code and integrity checks for IoT device updates are essential. A connected device's ability to receive OTA updates is advantageous, but it also poses security concerns. To reduce security risks for IoT devices, consider setting digital signatures on both the update code and the devices themselves. This ensures that only verified updates are used and that the code hasn't been altered in transit. OTA updates should also only be performed over encrypted communication channels, including any local connections between devices and the network.



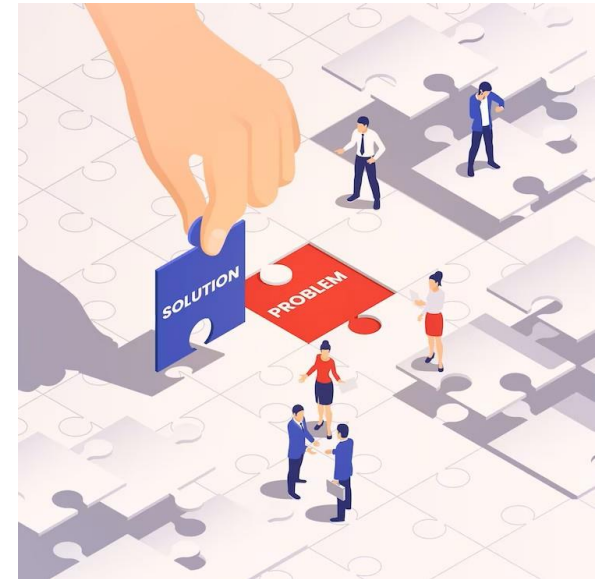
Scale with IoT deployments

As an IoT deployment expands, everything must scale accordingly. Networks must be able to adapt automatically and efficiently or else the system will quickly reach a breaking point that will cause IoT services to slow down or crash. When OTA updates add to the normal network traffic, it could lead to a breaking point. Each portion of a network has scalability features and costs that could quickly be affected with regular OTA updates. Organizations should implement an IoT platform that's able to scale quickly and efficiently from the very beginning. Platforms that store data in the cloud will enhance scalability and make OTA updates more efficient, while reducing bandwidth usage and costs.



Learn from consumers' challenges

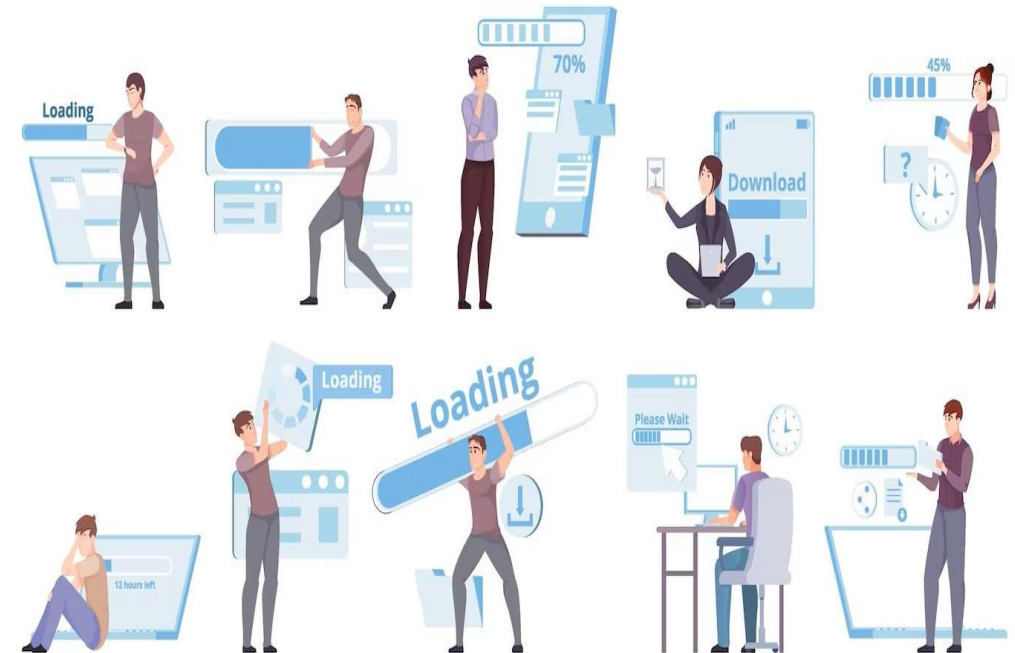
Organizations should consider the effect of IoT OTA updates on their customers. Consumers might use an IoT device in an unexpected way, creating unanticipated issues with updates. For example, consumers might install Wi-Fi components onto their IoT device that lead to a mismatch between what the update should do and how the device accepts the update. The IT pros sending updates might not know the device is Wi-Fi-capable and the change makes the update incompatible. Developing a relationship with customers will make it easier to stay in constant communication about how they're using their IoT devices. Organizations can then use customer feedback to develop updates that are agile and flexible enough to handle any curveballs.



Chapter 10

Risks of End-Of-Life OS

At some point every piece of software, including operating systems, reach a point called end of life. To limit the resources spent on supporting an OS, vendors set strict end of life dates for an operating system – after which official support, including fixes for security vulnerabilities, stops. **Relying on an end of life (EOL) operating system is a big risk, but it is nonetheless common for businesses to continue depending on an operating system that's no longer supported simply because the alternative of upgrading or changing to a different OS is too expensive or too inconvenient.**



Compliance and legal risks

With cybersecurity such a daunting problem, numerous legal and compliance requirements have stepped in to ensure that companies meet minimum data security standards to keep customers and clients safe.

It is common for these requirements to include a statement about official vendor support for software, and to deem the use of end of life software as non-compliant. Non compliance can lead to stiff penalties.

Companies that deploy an EOL operating system risk everything from fines through to losing their right to operate in their field. Where it can be proven that a successful attack was due to negligence, the use of end of life software being a prime example, companies may also be at the sharp end of legal proceedings because those impacted by the breach will seek redress.

Outdated and incompatible solutions

We said at the outset that technology moves quickly. Companies that harness the latest tech benefit in many ways. For example, offering better products and features to their customers, and a better experience for employees.

EOL software is by definition outdated software – missing a range of the latest features and benefits. Aside from leaving technology solutions in the slow lane, outdated software produces a further problem: a lack of compatibility. By relying on an end of life OS companies risk running into compatibility problems which will have a growing impact as time moves on.

Problems with reliability

One of the impacts of incompatible software is issues with reliability. Inevitably some elements of a technology solution will be upgraded: if the OS that the solution depends on is end of life, the operating system will be out of step with the rest of the solution.

This will lead to reliability problems as vendors will code with a certain functional expectation in mind, only for EOL software to lack the required functionality. In other words, a simple vendor update of a software component can break a solution due to an EOL operating system.



Growing costs

It is also worth noting that end of life software, including an EOL operating system, can represent a false economy. Yes, companies can save money by delaying an upgrade – but the maintenance of end of life software will grow with time as customized solutions need to be found to fix problems where vendor support no longer kicks in. Similarly, EOL operating systems can lead to expensive reliability problems that drive up costs beyond the savings realised from postponing an upgrade. Worse, as suggested above, outdated software can lead to compliance and legal problems that may lead to incredibly expensive fines.



Security risks

Last, we address the biggest issue with relying on end of life software: the significant security risks implied by using software that is no longer supported. By definition, EOL operating systems will not get security fixes and updates that protect users against known vulnerabilities.

Instead, these security risks will be known to the public, including hackers, but there will be no patch that can protect users against the risk where hackers decide to exploit it. A single critical bug that is not patched due to a lack of official support can lead to an expensive cybersecurity breach.



The firmware is a piece of code that resides in a non-volatile part of the device that allows and enables the device to perform the functions for which it was created. It consists of several components, such as the kernel, bootloader, filesystem and additional resources. In addition, the firmware makes various hardware components work properly.

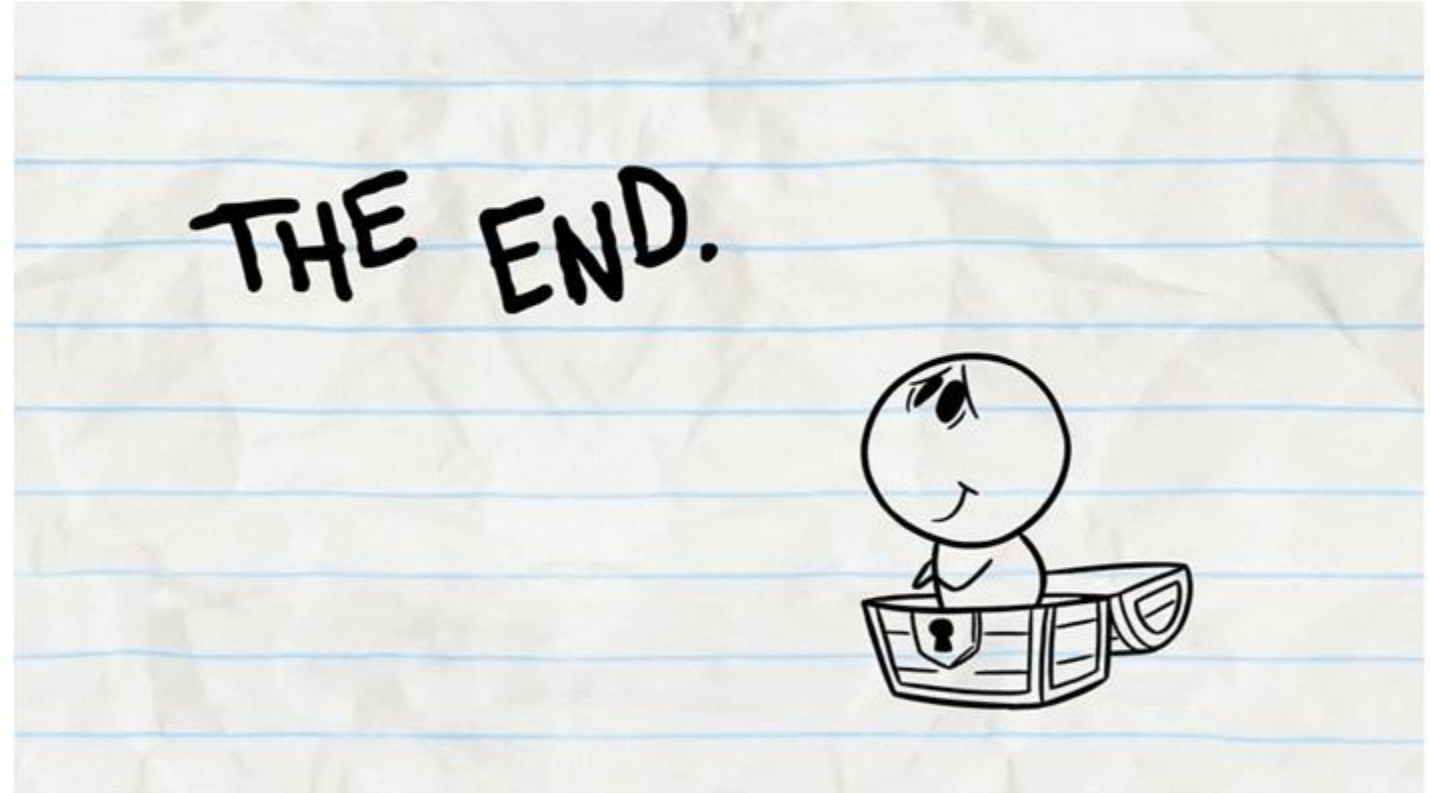
Knowing which different components are:

- **Bootloader:** It is responsible for numerous tasks such as the initialization of several critical hardware components and allocating the necessary resources.
- **Kernel:** It is one of the main components of the entire integrated device. Speaking at a very general level, a kernel is simply an intermediate layer between the hardware and software.
- **File System:** Is where all the individual files required for performance of the device are stored. This also includes components such as web servers and network services.



The End

Please feel free to contact me anytime if you have any questions



جامعة القادسية
EVERYONE'S SMART UNIVERSITY