

Guide to Patching

clickandprotect.co

Helping You Build Your Cyber Security






Table of Contents

| | | |
|-----------|------------------------------------|----------|
| 01 | What is patching? | Page 1 |
| 02 | What are the benefits of patching? | Page 1 |
| 03 | What should be patched | Page 2 |
| 04 | Patching isn't easy | Page 3 |
| 05 | How to manage patching | Page 4-7 |
| 06 | What if you can't patch? | Page 8-9 |
| 07 | Summary | Page 9 |
| 08 | Contact C&P | Page 10 |

1. What is patching?

Software isn't easy to produce. This is why it goes through extensive testing before being released.

However, inevitably—because coders and testers are human—there'll be small errors (known as bugs) in the code that were missed in testing. A few of those errors will leave security holes that need to be patched; some will mean that the software doesn't work exactly as intended. Once found, they'll need to be fixed.

Sometimes new threats are identified, that weren't taken into account when developing the code initially. And occasionally, updates that were intended to fix errors introduce new problems, which are identified later.

This is why software manufacturers issue new versions of their software, designed to fix errors and patch holes. These new versions are called updates, or patches.

They are not the same as upgrades. A software upgrade is a version of the software that supersedes the old version: it is intended as an improvement, offering new functionality, better speed or efficiency.

2. What are the benefits of patching?

Installing patches is a fundamental security hygiene practice. It can prevent security incidents and lower the impact of any incident that does happen.

Updates should be done regularly to:

- Reduce the security risk to your business by closing vulnerabilities.
- Make sure software is up to date, which may help with stability and therefore productivity.
- Add any available functionality updates, so you get the best out of your products.
- Ensure compliance. Regular and timely updates are often required as part of adherence to standards.

Updates should be done quickly, because once attackers know about software vulnerabilities, they may try to exploit them before the patches become available and are installed. Delay in installation obviously extends the period of time that your software is vulnerable to attack.



3. What should be patched?

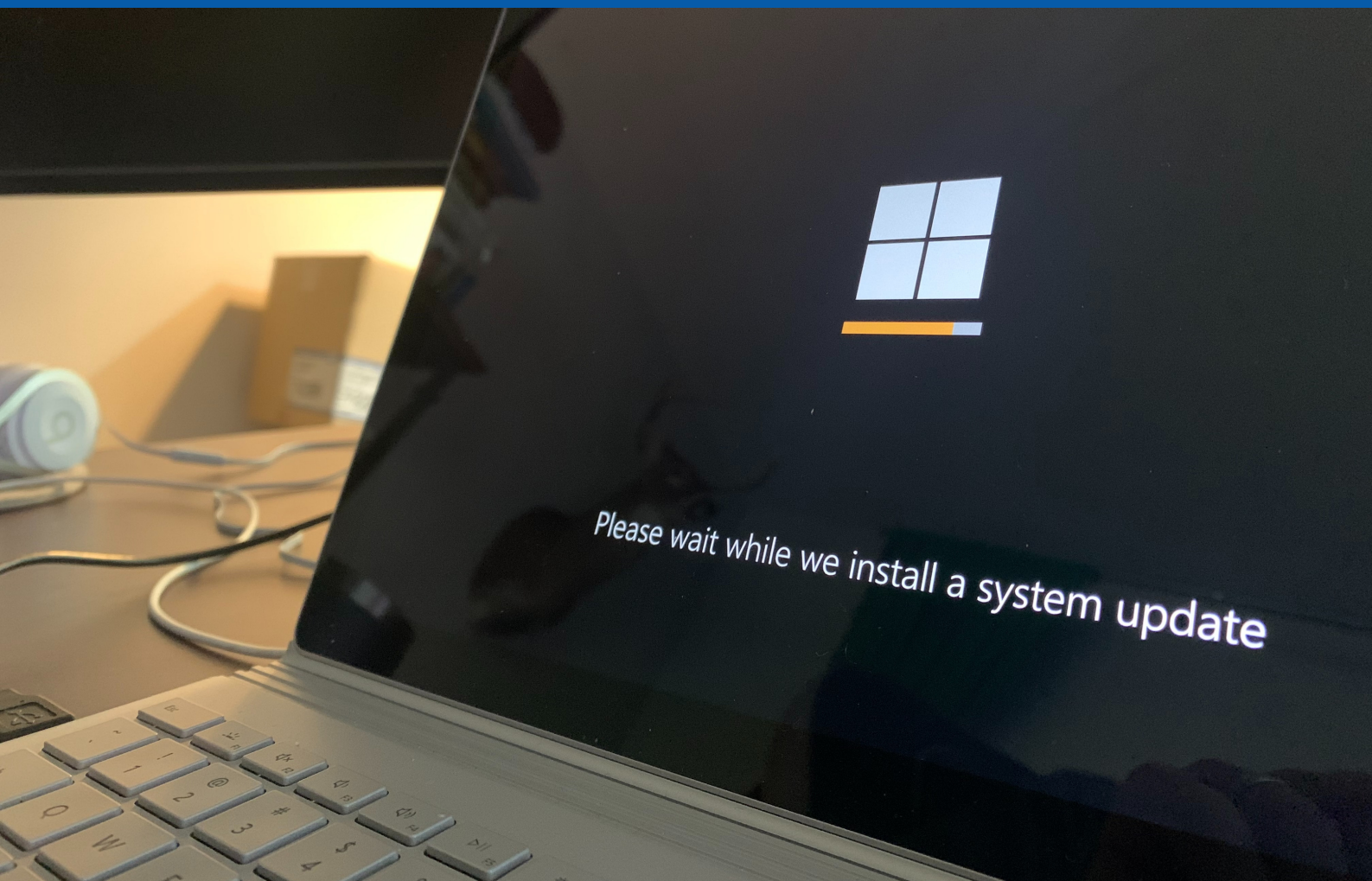
Any device that runs software may need regular updates.

This could include: your desktop, laptops, tablets and phones, your firewalls, hubs, switches and routers, other hardware, smart devices and no doubt more, depending on your industry.

It includes firmware, operating systems and software applications (commercial off-the-shelf, or in-house), your website and web applications. Think about everything on your network, and services provided to you through the cloud.

And, depending on your agreement with your provider, you may also be responsible for updating the server software that your email, cloud applications and websites run on, and the security software that protects it, and so on.

The first step in patch management is to know what you have that might need updates. It will probably be a longer list than you anticipate.



4. Patching isn't easy

It may seem obvious that you should implement updates when prompted, but it isn't always easy. Updating your own Windows laptop when you get an update alert is simple and straightforward—unless, of course, you are in the middle of a client presentation when an update starts.

But if you are doing much more than that, it takes both skill and time, and therefore money, and careful planning.

It may require some downtime for crucial business systems, and therefore will incur cost and inconvenience.

Also, for some companies, there are devices out in the field that might need to be recalled and brought in for patching, or which may require a field-trip for updates. Either of these requirements could delay updates.

You need to know what you have, in order to know what needs to be patched. For a small company, this might be simple, but for a large organisation, maintaining an up-to-date list of assets can be difficult. If there are a large number of assets to be updated, patching can become a fulltime job.

Some software is very hard to patch because the code can't be modified, or can't be modified quickly. This could be for one or more of many reasons, such as:

- The problem isn't in the code that you own, for example, it is in a commercial application. You have to wait for an official patch to be released.
- The problem isn't on assets that you own, though you do use them, so you are dependent on someone else doing the updates. These third parties might be vendors— or they might be employees with personal devices used for work, under your Bring Your Own Device (BYOD) policy.
- Fixes might take time to implement, because of the testing time required to make sure that making such a change doesn't negatively impact anything else.

- Fixes might be prohibitively expensive: perhaps it was poorly documented, or extremely complex, and you no longer have the skills in-house to implement a fix—or perhaps you just don't have the budget.
- The fix might be required to legacy code in an application: perhaps the vendor has gone out of business, or that version is no longer supported.
- Updates to your asset would require it to be recertified, which would take it out of use for some time.
- You are required to maintain an asset as is pending investigations of some kind.
- Updates to one asset require an updated version of another—but that one can't be updated for one of the reasons listed.
- You are running embedded systems, for which you may not have the source code.
- Your software is so old that patches are no longer available and/or it won't support newer versions of software.
- Your asset has no free memory to install a patch.
- There are known bugs in an update that would adversely impact your system.

Sometimes patching requires a reboot to take effect: this isn't always as straightforward as 'turning it off and on again', particularly if this is a production machine running 24/7.

And sometimes patching can fail. The patch installation may fail, or it might cause unexpected operational issues—particularly in complex systems—and need to be uninstalled, or patched in turn. Rolling back to the previous version of the system also adds time and costs to the process.

However, postponing patching gives attackers more time to take advantage of the vulnerability in your systems. If a patch fixes a security issue, it is likely that an attacker knows about the problem already. Perhaps by reverse engineering the patch to establish the issue with the code.

5. How to manage patching

At home, you can update your devices as needed, at a time that suits you.

For more complex business environments with many devices (and many stakeholders), there are patch management applications available, including patch management as a service.

These tools and services may not cover all software on all your devices, but should make the standard process work smoothly for many of them by:

- Scanning your network to identify missing patches.
- Downloading and applying patches to pre-specified groups of assets.
- Testing that patches were applied correctly.
- Reporting on patch status, rollout progress and any issues.

You'll need to decide what your strategy for patching will be, and plan the patching carefully.

Create a patch strategy

1. Identify your assets. All of them: hardware and software. Find out what software (and what software versions) you are using on which devices. Remember to include firmware and operating systems, not just applications. Don't forget to include shared libraries.

2. Run a vulnerability scan over your system to identify vulnerabilities.

3. Conduct a risk analysis of your assets. Which are the most critical to your organisation? Which are the most vulnerable?




```

the appear event when appropriate
k = function() {
  the element hidden?
  !t.is(':visible')) {
    //it became hidden
    t.appeared = false;
    return;

  the element inside the visible
  a = w.scrollLeft();
  b = w.scrollTop();
  o = t.offset();
  x = o.left;
  y = o.top;

  ax = settings.accX;
  ay = settings.accY;
  th = t.height();
  wh = w.height();
  tw = t.width();
  ww = w.width();

  (y + th + ay >= b &&
  y <= b + wh + ay &&
  x + tw + ax >= a &&
  x <= a + ww + ax) {

    //trigger the custom event
    if (!t.appeared) t.trigger

  } else {

    //it scrolled out of view
    t.appeared = false;
  }

create a modified fn with some
modifiedFn = function() {

  //mark the element as visible
  t.appeared = true;

  //is this supposed to happen
  if (settings.one) {

    //remove the check
    w.unbind('scroll', check)
    var i = $.inArray(check,
    if (i >= 0) $.fn.appear.c

  }

  //trigger the original fn
  fn.apply(this, arguments);

  bind the modified fn to the ele
  one) t.one('appear',
  settings.da

```

4. Review (or create) a patching policy, including documentation requirements. If relevant, make sure it covers updates for employee-owned devices.

5. Monitor to ensure you are aware of all available patches for your systems:

- Sign up to vendors mailing lists and notifications.
- Ensure these are sent to a central and dedicated location, so that they are not missed (especially emergency notifications, which would not arrive on a predictable schedule).
- Be aware of vulnerabilities identified in any third-party libraries and open-source software that you use.
- Consider subscribing to data feeds for vulnerabilities to receive updates.

6. Set up automated updates where possible. You could also automate scripts to test that patches were applied correctly. Automation will reduce the patching workload, but must be balanced against the risk of the automatic update causing a problem.

7. Prioritise the patches available based on your risk assessment, and plan to do the most important first.

8. Plan a routine patching schedule for assets with a regular release cycle. Remember: just because it is routine doesn't mean it isn't important. Don't postpone it. And don't forget about updating mobile devices too.

9. Plan patch releases, if that's appropriate:

- If patching requires downtime of a production environment, it would be better to take it down once, rather than repeatedly.
- Ensure that a bundle of patches in a release don't interfere with each other.
- Discuss with the business and risk owners. They need to understand the proposed changes, will know what the risks are for their area of work, and should sign-off on the patch release
- Try to avoid multiple updates to the same piece of software at the same time, if you are running behind on your patch schedule; later updates may have dependencies on earlier updates, and so they may have to be updated in a particular order.
- Don't forget to review patches that were excluded from previous patch releases, to check that these exceptions are still appropriate.

6. What if you can't patch?

Once you've identified those assets that you can't patch for some reason, you should identify other protections for those assets. Once you've decided how you'll protect them and developed an incident response plan just in case—you should monitor them carefully.

Other protections might include:

1. Restrict applications that can run on that asset to an approved list, so that other, uninvited software can't run.
2. Remove unnecessary connections. If that asset doesn't need to be connected to another asset, then removing that connection will reduce the risk.
3. Block removable media to stop data leakage and infection.
4. Segment the network to mitigate the risk—isolating the asset that can't be patched as much as possible.
5. Consider implementing hardware-based security, such as data diodes to enforce a one-way flow of data, if appropriate.
6. Consider virtual patching.

What is virtual patching?

Suppose that there is a known vulnerability due to a code error in one of your applications. A virtual patch is intended to prevent the vulnerability from being exploited, (even though the code has not been edited), by:

- Controlling the inputs to the application through a web application firewall, proxy or server plugin. It analyses traffic, and intercepts attacks so that malicious traffic doesn't reach the application.
- Or by controlling the outputs: either by blocking the entire outbound session (which might alert the attacker, or reduce the functionality of the system) or by redacting (or 'scrubbing') the output to prevent exposure of sensitive data.

Virtual patching has two goals: it minimises the time-to-fix (even if only with a temporary fix) and it reduces the organisation's exposure to the risk of attack.

Like the standard and emergency patching processes outlined above, there should be a consistent and repeatable process to virtual patching. This process would follow the same pattern.

Plan in advance for patching needs:

- For example, by developing policies, plans and procedures, or installing software modules that you might need to activate if patching.

Identify vulnerabilities:

- Via vendor announcements, public disclosure or, worst case—a security incident.

Determine the risk to your organisation of those vulnerabilities and whether virtual patching is appropriate and needed.

Prioritise the patching:

- Not forgetting to discuss the risks with the business owners, and ensuring that they understand this is a temporary and possibly incomplete fix.

Then create, install, test and document the patches.

Don't forget to consider updating the underlying code (which does, after all, still contain the problem) in future patch releases. If you do decide to patch the code at a later date, you can then determine whether you should remove the virtual patch.



7. Summary

Patching is important, and should be done regularly and as quickly as possible after the patch is released. There are exceptions; the risks associated with this should be carefully considered, and mitigations put in place.

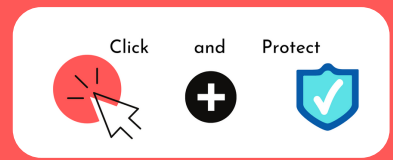
Patching becomes more difficult as the size of the system and the number of components grows. Proper processes and documentation should be put in place to manage it, and dedicated patch management applications considered.

So, in this guide, we have discussed what patching is, what should be patched and why. We've outlined some of the issues that people responsible for patching, can face.

We've described the steps involved in a patch management strategy, and in creating both a standard patch release and an emergency patch release.

Then we discussed what to do if you are unable to patch an asset, and described virtual patching, and the steps involved in setting up a virtual patch.

If you are wondering how best to implement patching in your organisation, call Click and Protect on 0113 733 6230 to find out how we can help.



clickandprotect.co
Helping You Build Your
Cyber Security

8. Contact C&P

Email: contactus@clickandprotect.co

Website: www.clickandprotect.co

Tel: 0113 733 6230

LinkedIn: Click and Protect

