Data protection

Protecting personal data in online services: learning from the mistakes of others

May 2014

ico.
Information Commissioner's Office

Contents

Introduction

1. The Data Protection Act 1998 (the DPA) is based around eight principles of 'good information handling'. These give people specific rights in relation to their personal information and place certain obligations on those organisations that are responsible for processing it.

2. An overview of the main provisions of the DPA can be found in The Guide to Data Protection.

3. This report relates to the seventh data protection principle and is intended to inform organisations about appropriate measures to safeguard personal data being processed by their computer systems. It is particularly relevant to organisations operating in an online environment.

4. The ICO recognises that there is a large amount of guidance already available in the wider field of information security, and this report is not intended as a comprehensive manual on the topic. Instead, it draws upon this existing knowledge while specifically focusing on the most significant threats to data protection. These are defined as those that have either resulted in a severe breach of the DPA or frequently occur in the ICO's casework. In almost all aspects the threats are not new and have been discussed in many information security publications. However, as these threats continue to exist, it is clear that some organisations still need to improve their the knowledge and understanding of effective information security principles.

5. The ICO assumes that the typical reader is someone who is either responsible for ensuring compliance with the DPA, or for managing computing infrastructure, or both. We assume this typical reader has a basic knowledge of computing but no particular information security knowledge. The level of detail provided here is therefore higher than in the ICO's related guidance A practical guide to IT security: ideal for small businesses.

6. This report is not aimed at experienced security professionals, who are already likely to have a good understanding of the issues covered here, and for whom more detailed guidance is widely available. However, the ICO's casework has shown that those generally responsible for IT security may also benefit from learning from the mistakes of others, to ensure the systems for which they are responsible do not suffer from similar problems.

7. This report describes eight frequently-arising computer security issues in an online environment that relate to data protection, together with a summary of good practice for how to guard against each issue. In many ICO data breach cases, the measures which could have prevented the breach or reduced the level of harm to individuals would have been simple to implement.

8. Some widespread computer security issues are not covered here, either because the overall data protection threat is relatively low (for instance, exposure of script debugging error messages) or because the evidence from the ICO's casework shows that they rarely feature in such cases, even if they commonly occur in other types of data security breaches (cross-site scripting falls into this category, for instance).

9. The scope of this report is also limited to frequently occurring information security threats in an online environment. This in no way reduces the need to consider other security issues such as the importance of encrypting laptop and mobile for instance.

Overview

- The ICO has identified eight important areas of computer security that have frequently arisen during investigations of data breaches. These areas are the focus of this report.

- The eight areas are:

    - Software updates
    - SQL injection
    - Unnecessary services
    - Decommissioning of software or services
    - Password storage
    - Configuration of SSL and TLS
    - Inappropriate locations for processing data
    - Default credentials

- For each area, the ICO provides advice on:

    - what data protection problems might be caused; and
    - good practice for avoiding those problems.

What the DPA says

10. "Data processing" is defined as follows in the Data Protection Act:

> "processing", in relation to information or data, means obtaining, recording or holding the information or data or carrying out any operation or set of operations on the information or data, including—
>
> (a)organisation, adaptation or alteration of the information or data,
>
> (b)retrieval, consultation or use of the information or data,
>
> (c)disclosure of the information or data by transmission, dissemination or otherwise making available, or
>
> (d)alignment, combination, blocking, erasure or destruction of the information or data;

11. Note that this definition is broad, and includes the action of simply storing data.

12. The seventh data protection principle states:

> Appropriate technical and organisational measures shall be taken against unauthorised or unlawful processing of personal data and against accidental loss or destruction of, or damage to, personal data.

13. It is important to note that, although many of the issues described in this report are easily avoidable through the measures to be described, an organisation must not assume that implementation of each recommendation will completely mitigate the risk of a breach or prevent each and every type of breach from occurring.

14. The interpretation of the seventh data protection principle (Part II of Schedule 1 of the DPA) also states:

> Having regard to the state of technological development and the cost of implementing any measures, the measures must ensure a level of security appropriate to—
>
> (a)the harm that might result from such unauthorised or unlawful processing or accidental loss, destruction or damage as are mentioned in the seventh principle, and.
>
> (b)the nature of the data to be protected.

15. This means that appropriate security measures will vary from case to case, depending on the circumstances.

16. The same section of the DPA also states:

> Where processing of personal data is carried out by a data processor on behalf of a data controller, the data controller must in order to comply with the seventh principle -
>
> (a) choose a data processor providing sufficient guarantees in respect of the technical and organisational security measures governing the processing to be carried out, and.
>
> (b) take reasonable steps to ensure compliance with those measures.

17. This report also aims therefore to better inform organisations ("data controllers") who are appointing IT service contractors ("data processors") and therefore need to ensure their contractors' compliance with the seventh principle.

Software security updates

18. Even the best, most secure and reliable software can have bugs in it, and some of these bugs will inevitably be security-related. Software developers will generally release updates (sometimes referred to as 'patches') for their software in order to fix these bugs.

19. If you do not apply security updates to a system's software, it will become progressively more vulnerable over time as more security flaws are discovered and methods for exploiting them become more widely-known. The same situation will arise when

the developers discontinue technical support for a software product, which normally means that no more security updates will be available.

20. Attackers typically run automated scans across a range of online services searching for un-patched, out-dated or otherwise vulnerable software to attack. It is therefore important that any software you use to process personal data is subject to an appropriate security updates policy.

21. There are many valid reasons why it may not be practical to apply security updates as soon as they are made available, such as:

    - an operational need to wait for a suitable maintenance period;
    - co-ordination with other necessary updates on related software;
    - the need to test updates before rolling them out to production systems; or
    - an assessment that a vulnerability does not affect the configuration used by the relevant systems.

22. However, it is still vital to define procedures that ensure updates do get applied within a reasonable time.

23. Additionally, you must also ensure that no relevant components are ignored. This is a common risk where responsibility for updates is split between multiple people, or where third-party libraries or frameworks are used.

Example

An organisation maintains a website which is hosted by a third party, and based on a content management system (CMS). The third party has clear responsibility for applying software updates to the web server and underlying operating system. The responsibility for updating the CMS installation lies with the organisation. However, the server-side scripting framework used by the CMS does not get updated at all since neither party has been assigned the responsibility. A vulnerability in the framework may allow an attacker access to the back-end database and to access, expose or retrieve personal data.

24. By default, you should only use supported software, ie software for which updates are still being provided. There are cases where it could be acceptable to use unsupported software, but you should strictly justify and limit its use. Do this on a case-by-case basis.

Example

A manufacturing company uses a computer to control a machine in one of its factories. The computer is running an operating system (OS) which will soon become unsupported.

The company justifies the continued use of the unsupported OS on the following grounds:

- The only personal data processed by the computer relates to the credentials of the supervisors that need to log in to it.
- There is currently only one software product available that can be used to control the machine, and it cannot be used with a supported OS.
- The machine is vital to the manufacturing process and would cause significant financial damage to the company if it were shut down.
- The computer is never connected to a network and has no wireless networking hardware.
- Access to the computer is physically restricted to supervisors.
- The company has a policy mandating extra security precautions for this machine, including restrictions on the use of removable media and using the computer for functions other than control of the manufacturing process.
- The company is actively investigating options for replacing the machine control software using supported components.

25. Don't forget to consider all your organisation's hardware assets, including laptops, mobile phones and other mobile devices. You will need to decide on the most appropriate method for applying updates, which might involve applying updates while the device is out of the office, or might involve returning devices to the office for regular updates.

26. You could breach the seventh data protection principle if you don’t define and adhere to an appropriate software updates

policy for systems that process personal data. Here are some considerations which can help your organisation define a software updates policy and keep to it:

- You can often group multiple systems together that have similar requirements, for example employees' desktop machines may all have the same updates policy. Whereas the central mail server will typically have different requirements and therefore a different policy

- Most modern software in common use will have automatic updates available in some form. As long as you trust the distributor of the software in question, you should consider whether it is practical to enable automatic updating. For example, areas which are not business critical to a particular organisation, perhaps work stations or standalone laptops, might be good candidates for automatic updates.

- Updates are normally better organised, where possible, using the operating system's native update process or package management system. For instance, it will normally be easier to enable automatic updates if you obtain software from an official source such as an app store or other software repository, instead of manually downloading and installing individual software components. Other benefits of this approach include improved security during the install process, and better assurance of compatibility of different software components.

- In some cases you can also streamline updates across multiple systems, for instance by maintaining a local repository of the latest software updates.

- If stability is an overriding concern, some operating systems allow only those updates marked as "important" or "security" to be applied, therefore reducing the overall number of changes made to the software.

- If necessary, you can prioritise updates according to severity of the security flaws that they fix. You should be cautious about deferring updates for supposedly lower risk vulnerabilities for too long though. Most obviously, even a low risk vulnerability still presents some risk, and the longer it is left unfixed, the greater the total risk exposure. Also, the 'low risk' classification may even change as more information is gained about the

vulnerability. Similarly, local circumstances may mean that what is generally considered 'low risk' presents a higher risk for a particular organisation.

- If for some reason you cannot apply a particular security update in practice, you could consider other security measures to mitigate the risk. This should be justified on a case-by-case basis, though.

- Your software updates policy does not have to be a separate document, and can be incorporated into your existing business process in the most suitable way for your organisation.

27. Remember that even a perfect updates policy will be ineffective unless it is adhered to in practice. You should have some way of ensuring that the relevant updates are being applied in accordance with your policy.

Software security updates – good practice summary:

- You should have a software updates policy in place for all software used for processing personal data. Ensure that **all** software components are covered by the policy, including operating systems, applications, libraries and development frameworks.
- There may be good reasons not to apply all available updates as soon as possible. Your policy can take into account these reasons.
- When there is no compelling reason to delay, you should apply security updates as soon as is practical.

SQL injection

28. SQL injection affects applications that pass user input to databases in an insecure manner. Typically this can occur in a publicly-available website that uses a database, in order to display information. The website might be a bespoke development or re-use other code, such as a content management system (CMS).

29. SQL injection flaws allow an attacker to inject database **instructions** using an input method which was only intended to

receive **information**. Typically this occurs via a form in a web page, for example a search box. Coding errors in the input method mean that the instructions are passed directly to the database. An attacker writes these instructions in a language called Structured Query Language (SQL), giving rise to the term 'SQL injection'.

30. In the most severe cases, an attacker could compromise the entire system by executing arbitrary code.

31. SQL injection is particularly relevant to data protection because:

    - The storage of personal data is very likely to be concentrated within some kind of database. This means that many breaches of the DPA have involved the compromise of a database.
    - Modern websites and web applications are very likely to be connected to a back-end database in order to provide dynamic content.
    - Most commonly-used databases use a variant of SQL.
    - An attacker does not need to compromise the database's authentication, since the application itself already has access to the database.
    - A potential attacker can automatically detect, and in many cases exploit, SQL injection flaws using commonly available tools.

32. In the following non-computer-based analogy for SQL injection, a person applies for a passport using a paper application form. In the space reserved for surname, they write *'Smith. Now tell me all the information you have about all the other passport applicants.'*. The officer processing this application form enters the name *'Smith'* into their system but then obeys the subsequent instruction and sends the applicant information about other passport applicants. Of course, this exploit would not generally succeed in the real world, since an officer is likely to realise that any instructions contained within the form should be ignored. Unfortunately the same is not true of many applications that are used to access databases.

Example
The following pseudocode illustrates a simple exploit as it might work on a system vulnerable to SQL injection:

```
sql_query_string = "SELECT * FROM users
WHERE lastname ='" + $user_lastname + "'"
database_query(sql_query_string);
```

Normally, the user might supply the last name "Smith" and this would pass the following instruction to the database:

```
SELECT * FROM users WHERE lastname ='Smith'
```

The database then returns a result set with all information about anyone with the last name Smith.

If however, a malicious attacker supplies the following input:

```
Smith' OR 'x'='x
```

This passes the following instruction to the database instead:

```
SELECT * FROM users WHERE lastname ='Smith' OR 'x'='x'
```

Since 'x'='x' is always true, the database returns all information about everyone in the users table.

33. SQL injection exploits have been a common theme across the computer-related data breaches that the ICO has investigated,. Prevention, detection and remediation of SQL injection vulnerabilities should therefore be a high priority for your organisation in comparison to other vulnerabilities.

Prevention and detection of SQL injection

34. Since SQL injection flaws are introduced in the source code of applications, it is important that you identify who is responsible for maintaining the source code of any application used. The processes for preventing introduction and detecting existing SQL injection flaws (or in fact any kind of flaw) will need to be slightly different depending on who is responsible. Generally, applications fall into two categories: externally-maintained applications; and bespoke, internally-maintained applications.

35. A typical externally-maintained application could be any software component that is used by one organisation but developed by another, including for instance:

- content management systems (CMS);
- forum software; or
- administrative web applications such as those that allow remote administration.

36. For externally-maintained web applications, you should consider the following:

- SQL injection flaws in a particular software product are normally announced in security advisories, meaning that you do not need to detect many potential vulnerabilities directly. The developers should release a software update or patch to fix the flaw, at the same time as, or shortly after the advisory.

- Although an external organisation may update the upstream application code, for example to fix an existing SQL injection flaw, you will still need to apply the update to the installation present within your organisation. This should normally be part of your software updates policy. (see "Software security updates")

- You may consider automated vulnerability assessments, often referred to as vulnerability 'scanning', for detection of some SQL injection vulnerabilities. Vulnerability assessment can typically identify simple examples of SQL injection as well as specific instances of known SQL injection flaws in common software products.

- You should consider regular penetration testing, if it is practical to do so.

37. Typical internally-maintained code could include:

- a company web application developed, hosted and used in-house;
- a public company website developed in-house but hosted externally; or
- a bespoke website developed by an outside company, but delivered to an organisation as is; in other words, without a support contract.

38. For internally-maintained applications, you should consider the following:

- Developers should work to a set of coding standards which include specific guidance and training on avoiding SQL injection.

- It is preferable to perform a review of the source code before it is put into production use. In this way, you can identify insecure database input methods early on. This can consist of peer-review to ensure adherence to an in-house style. An independent peer-review is also a possible option. In both cases, an audit trail is important to keep track of when the review took place, whether or not changes were made and who made them.

- Automated code review tools and vulnerability assessment can help identify code development issues in a bespoke application. However, you should not rely on them to identify all SQL injection problems on their own.

- Penetration testing is strongly advisable for any bespoke application that accesses a database. Preferably you should do this before the application goes into production use and as appropriate before the public release of subsequent updates.

- You should ensure that you have the capability to accept bug reports and fix security flaws in code you are responsible for.

Remediating existing SQL injection flaws

39. If you discover SQL injection vulnerabilities, you need to address them promptly and comprehensively. This will require changing the input method in the source code of the application.

40. One encouraging aspect for a developer responsible for fixing SQL injection flaws is that there is normally one accepted best practice available: namely, using the secure parameterised methods provided by the API in use. In different contexts these methods may be called by different names: for example, 'bound parameters', 'bind variables' or 'parameterised queries'. However, the effect is the same in that they are designed to make sure that **information** is never treated as a set of **instructions**.

41. Alternative methods may be used (with caution) if the API in question does not have parameterised methods available, namely:

- parameterised stored procedures;
- whitelisting of user input; or
- careful escaping of special characters.

42. Developers should avoid relying on:

- Any solution that uses a blacklisting approach. Such an approach can make exploits more difficult, but it can also reduce performance, and make code maintenance more difficult. Blacklisting can be useful as a short-term fix or for intrusion detection purposes. However, there are many ways in which an attacker could evade blacklisting.
- Any solution implemented client-side, for instance using javascript in a user's web browser. Such an approach can have benefits, but security is not one of them. Client-side scripting cannot be relied upon for security since the code would be entirely under the control of the user.

43. Finally, you should combine any attempt to fix a single SQL injection flaw with a review of the rest of the application's code to find any similar flaws. If an insecure input method has been used once, it is likely to have been re-used elsewhere, especially in code by the same author. For example, if your website's contact page has been found to be vulnerable to SQL injection, you should check all other locations accepting user input such as the search page or the login page.

SQL injection – good practice summary:

- Be aware of all of your assets that might be vulnerable to SQL injection. SQL injection can affect applications that pass user input into a database. This includes many modern websites and web applications.
- SQL injection presents a high risk of compromising significant amounts of personal data. You should consider it a high priority for prevention, detection and remediation.
- SQL injection results from coding flaws – so be sure you know who is responsible for developing and maintaining your code. It is these people who you will need to rely on to prevent SQL injection or fix SQL injection flaws if they are found. They will need guidance and training to understand the issue.
- Consider procuring independent security testing (penetration testing, vulnerability assessment, or code review, as appropriate) of the relevant sites or applications in order to identify code development issues, including SQL injection flaws. Do this before the application goes live. It is good practice to periodically test live applications.
- When remediating an SQL injection flaw, use parameterised queries where possible, and ensure that all similar input locations are also checked and remediated where applicable.

Unnecessary services

44. An important principle in network security is to only run the services that are absolutely necessary. This will reduce the number of ways an attacker might compromise systems on the network. If you have services which are publicly accessible and are not being actively used, you are exposing a range of potential attack vectors unnecessarily.

45. Whether a service is 'necessary' will depend on the business cost-benefit analysis of running each service. However, there are some cases where running a particular service will be clearly unnecessary. The following are examples of insecure services that can be exploited by an attacker and should be avoided wherever possible:

- telnet should be actively avoided because usernames and passwords are sent unencrypted. Most devices that use telnet can also use SSH, which offers a similar service but uses encryption.

- Plain FTP should be avoided for transfer of personal data or other confidential information; again because information, including usernames and passwords, is sent unencrypted. Remember too that FTP services can be configured to allow anonymous access which can also be indexed by internet search engines.

- Open mail (SMTP) relays: If you run a mail server which is acting as an open relay, spammers will be able to use it with no restrictions. This means your mail server is likely to be blacklisted by other mail servers, resulting in loss of mail service.

46. Additionally, there are many services that are suitable for use within a trusted network, but which should not be made available to the internet. Typically, these services should be contained within the relevant local area network (LAN) by using correctly configured firewalls. Examples include:

    - file- and print-sharing services such as:
        - those provided by default on Windows operating systems;
        - SAMBA;
        - Network File System (NFS) shares;
        - Common Unix Printing System (CUPS);
    - memcached;
    - direct database access (e.g. MySQL over TCP port 3306);
    - Universal Plug 'n' Play (UPnP);
    - Remote Procedure Calls (RPC);
    - Simple Network Management Protocol (SNMP); and
    - any service being used for internal development and testing only.

47. In cases where remote access to a specific service is necessary, you should consider options to reduce the risk of an attack, such as:

    - limiting access to trusted IP addresses only;

- providing the service via an encrypted method such as SSH tunnelling or by adopting a version that use Transport Layer Security (TLS); or

- ensuring that the service is segmented from others so that if it is compromised an attacker cannot gain access to other systems on your network.

48. A more general solution to allowing remote access would be to use a Virtual Private Network (VPN), which would allow remote users to be authenticated and also ensure that data is encrypted in transit. You must also consider the security of the access credentials, though, as these could be lost or stolen.

49. It is good practice to perform regular port-scanning of systems to discover any services running. This will help detect any unauthorised services that may have been introduced unexpectedly, for instance by an employee setting up a peer-to-peer file sharing system, or by an unauthorised piece of software which sets up a 'back door'. You can perform port-scanning both from an outsider's perspective, scanning the internet-accessible addresses in the network, and internally, from within the organisation's internal network.

50. Bear in mind that attackers will often attempt to perform port-scanning as well, so any authorised port scanning should be properly planned. For instance, administrators should be able to whitelist the source address of an authorised port scan in any IDS/ IPS that they have in place.

51. It's important that you periodically review the necessity of the services provided, as your network changes over time and decide whether any services can be restricted or completely decommissioned. Where you decide to decommission, you must do it thoroughly (see "Decommissioning of software or services").

Unnecessary services – good practice summary:

- Completely decommission any service that is not necessary.
- Avoid high risk services such as telnet.
- Ensure that services intended for local use only are not made publicly-available.
- Use periodic port-scanning to check for unnecessary services which have been inadvertently enabled.
- Maintain a list of which services should be made available. Periodically review the list to see whether any services have become unnecessary, and restrict or decommission them as appropriate.

Decommissioning of software or services

52. Organisations will sometimes be responsible for old computing services that may no longer be needed. This might include an inherited or legacy IT system but there may also be occasions when a service may be set up temporarily as a test or pilot. However, when an old or temporary service is no longer needed, you must decommission it thoroughly, otherwise it may continue to pose a risk. This risk could be direct, such as when a service is inadvertently left running and accessible. However, there may also be secondary risks resulting from failure to remove components such as binary executables or configuration files, which may help an attacker if they attempt a multi-layered attack.

53. The biggest danger with inadequate decommissioning is that because the organisation **believes** the service has been shut down, it becomes less likely that they will notice or manage the remaining risk.

54. This issue is illustrated further through the following examples. The common theme is lack of awareness of **all** aspects of a service and how they should be shut down.

Example
A company is responsible for administering the following two sites:

- sales-db.example.com
- apps.example.com

Both of these sites are hosted externally by a dedicated hosting provider, both on IP address 172.16.42.42[1]. Only sales-db.example.com collects and processes personal data, but both sites use the same back-end database. It has become clear that apps.example.com is no longer necessary for business use and in addition has not been adequately maintained for some time, so the company decides to decommission the site.

Since the company is in charge of their own DNS records, they simply remove apps.example.com from DNS so that users trying to visit the site normally using a browser will not be able to see the site. The company does not contact the hosting provider though, assuming that the site is now decommissioned.

However, an attacker can still visit IP address 172.16.42.42 and manually specify the HTTP header 'Host: apps.example.com' to gain access to the site. In this way, any flaws in the legacy site could still be exploited. Since sales-db.example.com shares the same database, some of these flaws may put the sales database at risk, and any personal data contained within it.

Even worse, with the assumption that the legacy site has now been decommissioned, it is now even less likely that the company will arrange for appropriate technical measures to be in place, such as security testing or software update procedures for the legacy site. This means that the likelihood of a compromise will increase over time and any such compromise may not be discovered until a significant time after the attack.

[1] The IP address 172.16.42.42 is used purely as an example here, since it falls within private address space as defined in RFC1918.

Example
A company wants to copy some large files from one computer to another, over the internet. The destination machine is intended to serve up a new corporate intranet website, which is not intended to be accessible from outside the organisation. Eventually, this intranet site will be used to process some personal data, but during development and setup no personal data is involved.

The transfer is a non-routine task so there is no established method to perform the transfer with. A system administrator therefore arranges for the transfer to take place by setting up an anonymous FTP service temporarily on the destination machine, which can be accessed via the internet. The files are transferred to the destination machine using this FTP service, with the FTP root being the same location as the files will be needed for the intranet site.

After the file transfer is complete, the system administrator continues setting up the new intranet site, but forgets about the temporary FTP server and leaves it running.

The intranet site is put into use and begins to process personal data. This personal data is now potentially at risk because of the anonymous FTP access which is still available. If an internet search engine were to index the FTP server, the personal data could be easily discovered through the use of that search engine.

Example

A company has historically used a webmail application to allow staff to gain remote access to their email. Because of a lack of available externally-facing IP addresses, the perimeter firewall is set up to use network address translation (NAT), port-forwarding requests on TCP port 443 directly to the webmail server.

The company decides that this external webmail service is no longer necessary and so the server is decommissioned by simply switching it off. However the NAT setup and firewall rules are not changed.

Later, an internal web application is set up to allow staff to access their personnel records. The old webmail server hardware is re-used to install a new operating system and set up the personnel web application on TCP port 443. The network DHCP server assigns this re-installed system the same IP address as the old webmail server had.

Although the company is not aware of it, the personnel application, which was intended only as an internal resource, is now accessible externally via the NAT and port-forwarding rules which are still in place.

Decommissioning – good practice summary:

- Be aware of all the components of a service so that you can make sure they are all decommissioned.
- Make a record of any temporary services which you will eventually need to disable.
- Thoroughly check that the decommissioning procedure has actually succeeded. Use systematic tools such as port
- scanners to do this where possible. Do not forget to arrange for proper disposal of any hardware, as appropriate (see the ICO guidance on IT disposal).

Password storage

55. Users' access credentials (eg a username and password or passphrase) are particularly valuable to attackers, for a number of different reasons:

    - An attacker who knows a user's credentials for a particular service can gain unauthorised access by impersonating that user.

    - A user may have re-used exactly the same username and password for other accounts relating to different services. This is especially likely if the username is an email address. In the worst case scenario, the user has re-used exactly the same password to access that email account.

    - A user's password may suggest a pattern which can be used by an attacker to guess other passwords. For instance, if an attacker discovers the password '`romeo1`', in January, then this would suggest that the same user might use '`romeo2`' when their password expires in February. Similar passwords might also be used for other services, such as '`romeo99`' or perhaps something more loosely related such as '`juliet22`'.

56. For these reasons, it is important that you manage credentials appropriately. In many contexts, you will already have appropriate password procedures in place (an example might be the authentication used by an already-installed operating system). However in some cases, you will have full responsibility for deciding how you handle passwords, for

example, where a bespoke web application uses a back-end database.

57. Secure handling of passwords is a final measure which reduces the adverse consequences of an otherwise successful attack which has defeated other security measures. However, it is important that password handling is considered before a serious attack occurs. Unfortunately the ICO's experience shows that poor password procedures often only come to light after a data breach.

58. If you do not have password security expertise available, you may want to consider using established third-party authentication services. Whether this is appropriate will depend on the sensitivity of the personal data in question, together with the level of trust placed in the third-party authentication provider.

The necessity of hashing

59. An important principle is that passwords should not be recoverable directly, in order to reduce the harm done if someone gains unauthorised access to the password information. This rules out the storage of passwords in plain text because these are immediately readable by a system administrator or a casual observer. It also means that encryption is not generally appropriate, since an encrypted value needs to be decrypted to retrieve the original password. This action of decryption requires access to a key, which must be securely managed. It also means that authorised staff (eg system or other administrators) who have access to the key may also be able to decrypt and access users' passwords. You should not underestimate the considerable compliance measures required for the safe and secure storage of such a key.

60. It is poor practice for an online service to be able to remind a user of their current password directly in plain text (eg by including it in an email). This would be a clear indication that the passwords are either stored in plain text or, at best, passwords are being encrypted but then decrypted. Either of these approaches would be insecure.

61. In practice, a procedure called 'hashing' should be used. A hash function is a **one-way** method which converts a password into a hashed value, often simply called the 'hash'. When a user first registers with a service and provides a password this is hashed and only this hash value is stored. When a user returns

and enters their password, the hash is freshly calculated then compared with the stored hash. If the two hashes match, then the user can be authenticated.

62. The one-way nature of hashes is key: if an attacker somehow obtains a list of hashes, they cannot directly work out what the passwords are, even if they know the particular hash function that was used.

63. An attacker can, however, guess passwords one after another, calculating the hash every time and checking for a match in the list of hashes (this can be referred to as an 'offline', 'hash cracking' or 'password cracking' attack). While hashing does not eliminate all probability that an attacker can discover the original password, if done appropriately, hashing makes password cracking attacks extremely time-consuming and therefore impractical.

64. The ultimate goal of hashing is that you will have adequate time to detect any password breach and take appropriate action (such as forcing a reset of compromised passwords) before the attacker has a realistic chance of correctly guessing any passwords.

The necessity of salting

65. It is important to use a technique called 'salting' to further guard against password cracking attacks.

66. A 'salt' in this context is a string of random data unique to each user. The salt is used by combining it with the user's password, then hashing the result. The salt is then generally stored alongside the hash in a database. When a user logs in to the service the stored salt and the supplied password are freshly combined and hashed. As in the unsalted method, the new hash and the stored hash are compared to determine if the user should be authenticated.

67. Even though salts will generally be available to any attacker who already has the related list of password hashes, using salts further increases the time and effort involved in mounting a password cracking attack. When brute-forcing more than one password, having a unique salt for each user prevents an attacker from successfully checking for matches across all the hashes, in turn slowing the overall attack down. To benefit from this, the only requirement of the salts is that they are unique for each user.

68. Salting also effectively increases the length and complexity of the value to be hashed, meaning that calculating all possible hashes in advance (in what's called a 'rainbow table') becomes much less feasible. The longer the salt, the less feasible it becomes to compute a rainbow table. A typical salt-length would be 128-bits, and this length is given as a minimum in NIST's "Recommendation for Password-Based Key Derivation".

69. A related benefit of unique salts is that it is not obvious from the list of hashes when two or more users have chosen exactly the same password.

The requirements of a password hash function

70. An important requirement of an appropriate hash function when used for passwords is that it is sufficiently time-consuming to compute. A hash function that takes a long time to compute can be quick enough for legitimate day-to-day use (ie a user performing a login will not notice a fraction of a second delay while the hash is calculated), yet it can severely slow down a password cracking attack. The key point to remember is that as computer hardware continually improves, the calculation of hashes becomes correspondingly faster. This means that, over time, some hash methods will become inappropriate for use with passwords because advances in computing hardware have made 'brute-force' attacks feasible. An offline brute-force attack simply involves computing the hashes of all possible passwords within a given scope.

71. An example of a hashing method that has been overtaken by advances in computing hardware is MD5. Since 1991, when MD5 was first developed, great improvements have been made which enable much faster calculation of hashes. This means that MD5 should not be used for hashing passwords. Appendix B contains the results of the ICO's own research into MD5 hash cracking, which shows how fast passwords can be obtained from MD5 hashes using even modest desktop computer hardware.

72. The SHA-1 hashing method, first introduced in 1995, is also widely regarded as inappropriate for password use, for example by NIST in the USA.

73. Password cracking has also become much faster in recent years through the use of Graphical Processing Units (GPUs) and even dedicated hardware. Cloud computing services can also be used on demand without any need for the attacker to buy

physical hardware. Skilled attackers would be likely to employ such resources when attempting an offline brute-force attack.

74. The time necessary to successfully guess users' passwords needs to be made significantly long since, according to Verizon's "2014 Data Breach Investigations Report", the majority of compromises are achieved in a matter of days (or less) while only a minority of data breaches are discovered in a similar timeframe. Combined with other data from the report, this indicates that the password hashing method should be strong enough to delay an offline brute-force attack for at least a matter of months.

75. Where you are responsible for choosing how to hash passwords, you should aim to use a password hashing scheme (technically known as a "key derivation function") that can arbitrarily increase the amount of computation required, in order to keep pace with future technological advances. This feature is often referred to as a "variable work factor". Examples include:

- PBKDF2
- bcrypt
- scrypt

76. Development in this field continues, so be aware that the list of available key derivation functions is open to change. The most important thing to remember is that you should review current good practice in key derivation functions during the design phase of the project so that you can choose a suitable option.

77. If using one of these key derivation functions is for some reason not currently possible or practical, then a simpler hashing algorithm might be acceptable. However, you should carefully consider the future risk of designing and using your own hashing scheme that will at some point become outdated.

78. A detailed assessment of cryptographic measures has been published by the European Union Agency for Network and Information Security (ENISA), part of which refers to password hashing. The report is entitled *Algorithms, Key Sizes and Parameters Report - 2013 Recommendations*. In section 3.3 it supports the above analysis, namely that:

    - MD5 is not appropriate; and
    - SHA-1 is acceptable for legacy systems but should not be designed into new systems.

Good password choice

79. Even with robust, salted hashing in place, there is still the threat of an attacker using a 'dictionary attack' to guess passwords that are known to be common. A good password hashing scheme can still be defeated if a weak password (ie a common one) is chosen by the user.

80. You should permit your users to choose a strong password, and where possible, actively encourage them to do so. Ways to increase the strength of a password include:

- creating a long password or passphrase;
- using a wide range of characters, such as:
  - uppercase letters;
  - lowercase letters;
  - numbers;
  - punctuation marks; and
  - other symbols.
- avoiding the use of dictionary words where possible;
- avoiding simple substitutions such as 'p4$$w0rd'; and
- avoiding the use of patterns derived from the physical keyboard layout (e.g. 'qwerty' or '1qaz2wsx'), since these are extremely common among users who do not use a latin alphabet in their mother tongue.

81. You should advise your users to use techniques such as those listed above. If well designed, a 'strength meter' can be used when setting passwords, to give the user immediate feedback. You may also wish to consider blacklisting the most common passwords.

82. In some cases, if a common password has been chosen, an attacker may not even need to mount their own password cracking attack: instead they could simply search the web for the hashed value to try and find the corresponding password.

83. Do not unnecessarily limit the permissible character set or the maximum length since this would simply reduce the time an attacker would need to mount a successful brute force attack. In any case, there is often no obvious benefit of limiting password length when password hashing is correctly

implemented, since the hashed values will all be the same, known length, requiring the same storage space.

Password hashing – good practice summary:

- Don’t store passwords in plain text, nor in decryptable form.
- Use a hash function. Only store the hashed values.
- The hash function should have appropriate strength to make offline brute-force attacks extremely impractical.
- Use salting to make offline brute-force attacks less effective.
- Periodically review the strength of the hash function and keep up to date with advances in computing power. The best way of achieving this is to use a password hashing scheme with a configurable work factor.
- Use a combination of password strength requirements and user-education to ensure that attackers can't simply guess common passwords.
- Have a plan of action in case of a password breach. This should include how to reset users' passwords in bulk and how to notify them of what has happened and what they need to do about it.

Configuration of SSL or TLS

84. Secure Sockets Layer (SSL) and Transport Layer Security (TLS) are closely related encryption schemes used for ensuring secure communications across the internet. For example, visiting an `https://` website means the site is using either SSL or TLS. In practice, the single term 'SSL' is often used loosely to signify either or both of SSL and TLS, even though TLS has been in existence since 1999 and is now widely used and supported.

85. A connection between two systems using SSL or TLS is designed to provide two assurances:

    - The communication is encrypted.
    - The identity of one or both of the endpoints can be trusted.

86. Misconfiguring an SSL or TLS service will cause one or both of these assurances not to be guaranteed. However, both assurances are required to create a trusted connection so that personal data or other sensitive information can be securely transferred.

Assurance of encryption

87. Failure to provide the first assurance (encryption) means that any sensitive information transmitted will be viewable via any computer system on the route between the two systems.

88. Most obviously, it is important to verify that SSL or TLS is running as expected. For instance, just because a website is made available over the default port for an SSL-enabled web server (TCP port 443) does not mean it is using SSL or TLS automatically. SSL or TLS must be specifically enabled.

89. The simplest way for an attacker to defeat the encryption provided by an SSL or TLS service is if the same data is already available via a non-encrypted method. This can happen, for example, if information is made available on an `https://` website but is also made available over a plain `http://` connection. You should have a clear concept of which information needs to be encrypted and which does not, and apply the use of SSL or TLS as appropriate. To reduce complexity, you may also wish to consider using SSL or TLS throughout your entire domain.

90. In the case of a website, any included content such as images, javascript or CSS should also be provided over SSL or TLS. Failure to do so risks compromising the security of the connection and will generally result in 'mixed content' warnings in users' browsers.

91. It is also poor practice to allow users to remain logged in to a website or other web application if they navigate from an `https://` page to an `http://` page. This could allow an attacker to read a user's session cookie, which in turn would let an attacker gain access as if they were a logged-in user.

92. The particular settings enabled for any SSL or TLS connection need to ensure adequate encryption:

    - Do not use SSL version 2, as this is insecure. At a minimum, use SSL version 3, but preferably use TLS with the latest version possible. TLS 1.2 is current at the

time of writing, and there is now widespread support for it amongst modern browsers.

- Do not enable any weak ciphers. As a minimum, only enable ciphers with 128-bit strength or greater.
- Avoid any ciphers with known weaknesses in their design, such as RC4.
- Do not enable ciphers that provide no encryption or no authentication ('null ciphers').

93. Other organisations have provided current, detailed, practical advice on which protocols and ciphers to enable, for instance at bettercrypto.org .

Assurance of identity

94. Failure to provide identity assurance opens users up to other types of exploit. Identity assurance in SSL and TLS is based around the concept of a digital certificate.

95. Users of any SSL or TLS service will receive a digital certificate from the server. It is possible for the server to require the user to provide a client certificate too (for instance when connecting to a VPN). However, the most common situation is for only the server to provide a certificate, leaving users to be authenticated using another method such as login with a username and password.

96. The server certificate will only provide the required assurance of identity if it is valid. Any one of the following would make the certificate invalid:

- Common name(s) on the certificate do not match the name of the site.
- The certificate has expired or is not yet valid.
- The certificate has not been signed by a trusted Certificate Authority.
- The certificate has been revoked.

97. A valid digital certificate is designed to assure the user that the organisation has satisfied a Certificate Authority that it is legitimately in control of the domain name(s) for which the certificate is issued. Extended Validation (EV) certificates are also available from Certificate Authorities to provide a higher level of assurance.

98. When SSL or TLS is used for multiple name-based virtual web servers, you should ensure that **all** `https://` sites served by the same system are covered by a valid certificate. This can be achieved by using:
    - a wildcard certificate (this method cannot be used with EV certificates);
    - multiple domain names in the certificate's 'subjectAltName' field; or
    - TLS Server Name Indication together with multiple certificates.

99. An invalid certificate will normally cause the user's software (for example, a web browser) to provide a warning that the connection cannot be considered secure. These warnings can often be ignored by users, but it is not good practice to train users to do so.

100. Where SSL or TLS is used internally within an organisation, it can often impose an unnecessary financial burden to purchase certificates from an external Certificate Authority. In this situation, it is good practice to set up your own internal Certificate Authority, which can then be treated as trusted by your corporate systems.

Exploiting lack of identity assurance

101. The most obvious problem is that a service without any assurance of identity misses out on an opportunity to provide users with additional trust in the service. If your organisation does not provide adequate identity assurance, users have one less way of distinguishing between your trustworthy service, and any other service which could be engaging in fraudulent behaviour.

102. Lack of identity assurance is a more direct problem when exploited using a 'man-in-the-middle' attack. A man-in-the-middle attacker impersonates the SSL or TLS service that a user expects to see. If the regular users of a legitimate service become accustomed to seeing security warnings and having to ignore them in order to use the service, they will be likely to do the same and ignore security warnings about an impostor. This means the user will potentially set up an untrusted connection despite SSL or TLS being in use and the communication being encrypted between the user and the attacker.

103. The man-in-the-middle attacker then forwards the communication onto the legitimate service and acts as a go-

between, observing the communication unencrypted on its way past (hence the name 'man-in-the-middle'). In this way, both the user and the legitimate service are potentially unaware that the connection is compromised, yet the attacker can easily obtain unencrypted information.

Example

1. A company maintains their main website at `https://www.example.com`
2. The company also maintains a login page at `https://login.example.com` , which uses TLS.
3. `https://login.example.com` has an invalid certificate (because the common name given on the certificate,'www.example.com' does not match).
4. Users logging-in routinely have to ignore the certificate security warnings that their browsers show when they try to log in.
5. An attacker gains themselves a position on the same network as one of the users. They use ARP spoofing to act as a man-in-the-middle, forwarding traffic between the user and the company's server.
6. The user then attempts to log in at `https://login.example.com` . The attacker notices this attempt, and tries to capture the user's credentials.
7. The nature of the TLS connection means that even though the man-in-the-middle would be able to read the encrypted traffic, they wouldn't be able to decrypt the communication between the user and `https://login.example.com` .
8. To get round this, the attacker discards the server's valid certificate and in its place supplies a fake certificate to the user. The attacker knows the private key corresponding to this fake certificate, and so is able to decrypt any TLS connection set up using it.
9. However, the certificate is invalid (because it was not signed by a trusted Certificate Authority) and gives rise to a certificate security warning in the user's browser.
10. The user ignores the security warning provided by their browser, since it looks almost identical to the one they regularly have to click when logging in.
11. The attacker can now decrypt the communication between the user and `https://login.example.com` , and gain access to any information exchanged, which would include the user's login credentials.

Configuration of SSL or TLS – good practice summary:

- Ensure that personal data (and sensitive information generally) is transferred using SSL or TLS where appropriate.
- Consider using SSL or TLS for all data transfer in order to reduce complexity. Remember that in the case of a website, any included content such as images, javascript or CSS should also be provided over SSL or TLS in order to avoid 'mixed content' warnings.
- Ensure that SSL or TLS is set up to provide encryption of adequate strength.
- Ensure that **every** SSL or TLS service uses a valid certificate, and schedule renewal of all certificates before they expire to ensure the services remain secure.
- Consider obtaining an Extended Validation (EV) certificate if assurance of identity is of particular importance.
- Do not encourage users to ignore SSL or TLS security warnings.

Inappropriate locations for processing data

104. A large number of data breaches investigated by the ICO involve personal data being processed in an inappropriate location. This can put the personal data at risk unnecessarily and threaten its confidentiality or integrity, or both.

105. Remembering that the DPA defines "processing" to include storage, there are typically two main types of cause of this problem:

    - Poor security architecture, meaning that it isn't clear where and how personal data should be processed.
    - Inadvertently storing personal data in a publicly-accessible location (even though there may be a well-designed environment with appropriate security architecture in place.)

Security architecture

106. Poor design of systems and networks can contribute to compromise of personal data. On the other hand, having well-designed systems and networks can provide clarity as to where personal data should be processed internally, and help prevent it being leaked to inappropriate locations.

107. An important principle is segregation of production environments from development or testing environments. The most significant reasons for this are:

    - While developing changes, your production systems (and the business functions they provide) need not be affected. This is important for business availability as well as security reasons.
    - A separate testing environment gives advanced notice of many operational problems together with the opportunity to fix them without affecting the production systems.
    - Development and testing should be performed with example data, rather than personal data. This avoids personal data being unnecessarily put at risk during these phases.

108. Similar segmentation according to function can further improve security and also make security measures scalable. A well-known example of this kind of measure is the concept of a DMZ (standing for 'de-militarised zone') where a separate network zone is segregated for the purpose of providing external services to the internet. The systems within a DMZ provide more open access to services, and therefore are intentionally exposed to a greater security risk than those systems within the main body of the internal network, where typically no direct access from the internet will be intended. An important point here is that access is typically controlled by firewalls and routers, and granted on the basis of network zones instead of individual IP addresses.

109. In the same way, you can use internal routers/ firewalls to control access between internal network segments. This kind of segmentation can be made to match your policies for processing personal data.

Example
An organisation might have chosen to restrict processing of certain personal data so that it must be done only by authorised members of a particular department, for example Human Resources. It might then be appropriate to give the Human Resources department a designated network segment within which the personal data must be restricted. Firewall policies can then be put in place, for instance to deny access to the systems storing the personal data if the request originates from outside the designated Human Resources network segment.

110. Using segmentation and applying policies according to network zones rather than individual systems also allows for a more scalable approach. You can add or remove systems from the different network zones without necessarily having to change firewall policies.

111. Keeping an inventory of systems and where they are located within the network architecture can help prevent systems becoming 'orphaned' so that they are no longer maintained or monitored. Such an inventory could also discourage haphazard development of the network architecture, forcing system administrators to choose an appropriate location for each new component as it is added.

112. It is also important to consider all situations where data may be co-located, including if you are using a backup or failover device to store data from multiple systems.

113. Network security architecture is also relevant to another aspect of the seventh data protection principle, namely "accidental loss or destruction of, or damage to, personal data". You can guard against this by ensuring redundancy and diversity are appropriately included within the network.

114. A simple example would be the common practice of using both on-site and off-site backups. Regular on-site backups guard against data loss, destruction or damage to a subset of the data on-site, for example due to a hard drive failure or user error. Off-site backups may be done less frequently, and may need more effort to restore from, but guard against larger-scale common failure modes such as an on-site fire, virus outbreak or

insider attack. Network architecture will clearly be important in both cases, for example:

- Backups should be designed to mitigate specific risks. For instance, mitigate the consequences of a hard disk failure, the on-site backup must be saved on a separate hard disk. This would be particularly important to consider if the network environment included network-mounted file systems.
- An off-site backup will not be appropriate if it is exposed to a failure mode in common with the on-site machines, for example if it is stored in a ground floor location in the same flood plain as the main on-site facility.

Security architecture – good practice summary:

- Ensure testing or staging environments are segregated from the production environment.
- Consider segmenting your network according to function and in accordance with your data protection policies.
- Ensure your network architecture accounts for functions such as backups and business continuity in general.

Storing personal data in a widely-accessible location

115. Many breaches of the DPA in the physical world have had the same straightforward cause, namely: inadvertently leaving a copy of personal data in an accessible place where unauthorised people could access it. The same problem exists in the world of computers and networks. This particular problem can also occur even when there is clear security architecture in place.

116. There are three mistakes that can be made, either separately or together:

    - failure to realise that the storage place is widely-accessible;
    - failure to realise the personal nature of the data in the first place;
    - administrative error.

117. All three of these mistakes will need to be addressed using a combination of policies, training and monitoring. However, technical measures can be taken in support of these

organisational measures, to provide defence in depth and eliminate reliance on a single control.

118. For instance, failure to realise that a storage place is widely-accessible could be avoided in some cases by regular port-scanning for unexpected services or devices. An example of this would be a web server made available to the internet even though access is intended to be restricted to internal staff use. An external port scan, properly performed and then acted upon, could alert system administrators to the problem and allow them to fix it. (see "Unnecessary services")

119. Another extremely common example involves a supposedly hidden directory within a web server, for instance the website administrator for `www.example.com` might create a directory located at `http://www.example.com/private/` . The most common mistake is to assume that just because no links are provided to this directory, access is somehow restricted. However, without specific access restrictions, an attacker could easily access the directory if they could guess the directory name. In many cases, guessing a directory name is made easier when common names are chosen, such as `/private/` in the example above. Other examples of easily guessable common directory names are:

`/cv/`	`/admin/`	`/uploads/`	`/data/`
`/images/`	`/docs/`	`/icons/`	`/service/`
`/db/`	`/forum/`	`/search/`	`/cms/`

120. This list does not include any examples of the many application-specific directory names which would also be well-known to any attacker.

121. Remember though that the important point here is that a directory of any name can be accessed if there are no proper access restrictions in place. Choosing an easily-guessable directory name simply makes an attacker's job easier.

122. An extra risk is posed if you allow directory browsing without being aware of the consequences. Apart from any access control issues, this needlessly leaks information about which files are present on the webserver and may well help an attacker to focus their efforts.

123. In many cases the files may not even be accessed with malicious intent. If a file is publicly accessible on the internet without additional access controls, then there is no barrier to anyone accessing these files, including internet search engines. The web crawlers of internet search engines continually search the web (and in some cases FTP servers) for content to index and cache. If a web crawler encounters a web server with directory browsing enabled, the entire contents can be indexed within minutes and therefore its contents made searchable via the search engine's user-friendly interface. Local website search facilities could also present a risk if they expose internal documents which should be private.

124. Sometimes other vulnerabilities, which by themselves are perhaps lower risk, will expose locations that are wrongly thought to be hidden.

Example

A company stores personal data within a sub-directory of a webserver:

```
https://www.example.com/private/
```

The company does not want the contents of this directory made public, and so puts the following into a robots.txt file on the webserver:

```
User-agent: *
Disallow: /private/
```

The company wrongly thinks that the personal data is now inaccessible.

However, a malicious attacker would not in any way be obliged to obey the instructions in robots.txt, and this file serves no security function.

In the absence of any other measures, the data in the `/private/` directory is publicly accessible.

Additionally, the information in robots.txt has revealed to any potential attacker a location where personal data may well be stored. This allows their attack to be better focused.

Furthermore, accidental deletion of the robots.txt file would no longer signal to web crawlers that the contents should not be indexed.

125. There are many other information leakage vulnerabilities which would have the same effect of disclosing the names of supposedly hidden directories. While they may be low risk on their own, it is good practice to pay attention to them in case they are used as starting points for further attacks.

126. A similar situation occurs when using a unique URL parameter as part of a dynamic website. For example, if the URL `http://www.example.com/printOrder.do?orderID=443` displays the receipt for an order without any additional access controls (eg requiring the customer to be logged in) then if the website user edits the URL manually (eg `printOrder.do?orderID=442`) they will be able to access the orders (and therefore the personal data) of potentially all previous customers.

127. Deciding what constitutes an appropriate storage place is ultimately an issue of policy, because so much depends on the nature of the personal data and how organisations choose to manage this risk. However technical measures can help ensure compliance with policy. For example, if an organisation needs to process particularly sensitive information, they may choose to set up a network which is isolated from any other network, to avoid an unintended data transfer. A similar effect could be achieved by using internal firewall policies. Remember that in most cases, there will be an element of human intervention in the processing of personal data, and so any technical measures taken should not be absolutely relied upon: policy and procedure will still be necessary so that staff know what is expected of them.

128. Careful security architecture can also provide awareness of personal data and where and how it should be stored (see "Security architecture"). However in general, this awareness will ultimately depend on good training and appropriate policies. Training will be more effective if it is tailored to different roles in the organisation. For example, an office worker might need to know how to recognise personal data, where to save electronic documents to, and how to comply with a clear desk policy. In contrast, a system administrator might need much more detailed training since they will be responsible for larger amounts of personal data.

Accessible locations – good practice summary:

- Make sure you have policies for how, when and where personal data will be processed.
- Consider all the services you are running, how they are accessible, and whether they comply with your policies.
- In particular, ensure any web servers are exposing only the intended content. Where necessary, apply specific access restrictions. Do not rely merely on obscurity to prevent access.

Default credentials

129. Many software components are distributed with default credentials provided, typically a username and password. This can make distribution, installation and set-up simpler, but also poses a security risk.

130. Some general examples where default credentials might be provided are:
    - administration interface to a wireless router or firewall;
    - guest account for a content management system (CMS);
    - administration account for a database; or
    - default accounts for an operating system.

131. You should change any default passwords as soon as possible, normally before development or testing begins, and certainly before the relevant software component is put into production. Failure to do so means that any personal data processed using that system or service could be at risk from unauthorised access.

132. Remember that if an attacker can gain an indication of what type of system or service they are trying to access, one of their first considerations will be "is there a default password?" If there is a default password, it can normally be easily found using a simple web search. Even in cases where the type of service or system is unknown, an attacker can simply try various combinations of common default credentials. This also applies to services considered secure such as SSH which could

be used to access a server which has not changed the default root password and not disabled root login via SSH.

133. Remember that some software and operating systems could have guest or demonstration accounts enabled, which may have their own default credentials.

134. When changing default access credentials you should also follow good practice on password choice (see "Good password choice"). The benefit of changing a default password will be minimal if the new password is weak (ie common and easy to guess). Even worse, the benefit will be negligible if, say, the new password chosen is also a common default password for a different service or system.

135. Access credentials should not be hard-coded. In some contexts hard-coding is obviously inappropriate because the credentials can be obtained by simply viewing the source code (eg database credentials revealed in the source code of a web page, perhaps within javascript or HTML comments). More generally, though, hard-coding of credentials poses a security risk because it involves storing a password, which then becomes a target for attackers. This target is all the more vulnerable if credentials are stored in plain text.

136. Hard-coding is also inappropriate because it leads to poor maintainability. If a hard-coded password is breached so that unauthorised people have access to it, changing this password requires the developer to make changes to the code and then roll out those changes, potentially with testing and staging phases in between. If the credentials are not hard-coded, there will normally be a standard password-reset process which can be followed.

Default credentials – good practice summary:

- Change any default credentials as soon as possible.
- When changing default credentials, remember to follow good practice on password choice.
- Ensure that credentials are not hard-coded into any of your software.
- Ensure that credentials are not transmitted in plain text.

Other considerations

137. The eight areas covered in this report are only part of the wider field of information security. You should not ignore other measures that guard against other issues.

138. Further relevant ICO guidance includes:

- A practical guide to IT security: ideal for small businesses
- IT asset disposal guidance
- Guidance on the use of cloud computing
- Bring Your Own Device (BYOD) guidance
- Guidance for app developers "Privacy in mobile apps"
- Guidance on data security breach management
- Notification of data security breaches to the Information Commissioner's Office

139. As technology advances, it is highly likely that new or different threats will be commonly seen in the ICO caseload. This document should not be considered an exhaustive or complete list of potential threats. Any organisation operating an IT environments must ensure that appropriate technical and organisational measures are in place to protect personal data. These must be monitored and maintained over time to remain effective.

140. Further guidance on the wider field of information security can be obtained from organisations external to the ICO and includes:

- CESG's "Ten Steps to Cyber Security"

- The Open Web Application Security Project (OWASP)

141. If you need any more information about any other aspect of data protection, please contact the ICO via the website at www.ico.org.uk.

Appendix A – Glossary of abbreviations

API	Application Programming Interface
CMS	Content Management System
CSS	Cascading Style Sheets
DPA	Data Protection Act
ENISA	European Union Agency for Network and Information Security (originally: European Network and Information Security Agency)
FTP	File Transfer Protocol
IDS	Intrusion Detection System
IPS	Intrusion Protection System
LAN	Local Area Network
NIST	National Institute of Standards and Technology (USA)
OS	Operating System
SQL	Structured Query Language
SSH	Secure Shell
SSL	Secure Sockets Layer
TLS	Transport Layer Security
VPN	Virtual Private Network

Appendix B – Cracking MD5-hashed passwords: results

Time taken to compute all possible MD5 hash values for various character sets and password lengths

	Password Length (characters)										
Character set (number of different characters)	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
numbers only (10)	<1s	<1s	<1s	<1s	<1s	<1s	<1s	4s	41s	6min 47s	1h 10min
lower case only (26)	<1s	<1s	<1s	<1s	1s	11s	5min 16s	2h 15min	2day 13h		
lower case & numbers (36)	<1s	<1s	<1s	<1s	2s	1min 22s	50min 21s	1day 6h			
lower and upper case (52)	<1s	<1s	<1s	<1s	14s	12min 11s	11h 14min				
Lower & upper case & numbers (62)	<1s	<1s	<1s	1s	32s	33min 28s	1day 14h				
lower & upper case & numbers & special characters (95)	<1s	<1s	<1s	3s	4min 47s	7h 38min					

Hardware used:

- Intel® Core™ i3-2120 CPU @ 3.30GHz × 4
- 4GB RAM

Software used:

- hashcat 0.44
- 32-bit Ubuntu 12.04 LTS