1. You have a well-defined catalogue of the data your service holds. You know why the data is held. You understand the impact of theft or loss of integrity.

To make intelligent decisions about the design and operation of your service you need an accurate understanding of what would happen if data were lost or compromised. This knowledge will also guide your actions in the event of a breach.

Reputational damage and loss of public confidence are obvious repercussions of any incident, but more subtle second-order implications are often encountered. These could include the possibility for future attacks against your system, so it's important to take the time to develop a full understanding of potential impact.

As your system's requirements change, you should evaluate the necessity of the data you hold. If, in addition to knowing what data you hold you are clear exactly *why* you hold it, this process will be greatly simplified.

2. You know that only necessary data is captured and held by the service. You regularly check this remains the case. Records which are no longer required are removed.

Breaches are worse than they otherwise would be when superfluous information is collected.

Sometimes the problem is simple: You're just not removing old records when they're no longer required. Sometimes it's more complicated.

Here are some example scenarios to be avoided:

- Not removing partially completed applications
- Collecting unnecessary metadata relating to users
- Repeatedly collecting the same data at different stages of an interaction
- Inadvertently caching information in temporary data stores
- Recording personal data in debug logs

3. You have an accurate understanding of unmitigated vulnerabilities present in the service and are actively managing them.

Vulnerabilities could exist in any of the technologies underpinning your service. Operating systems, networking equipment, development frameworks and software packages, all are at risk.

Tracking the versions of these, how they are configured, and cross-referencing with vulnerability notifications will help you understand your exposure to publicly known attacks. You will then need to classify and prioritise remediation according to the potential impact to your service.

Failure to mitigate these vulnerabilities will make your system easier to compromise and exacerbate the impact of any successful attack.

4. All users with access to your data are individually known and referenced. Users are only ever granted the level of access required to perform their job. Access and privileges are removed as soon as they are no longer required.

In order to control and audit access to data you need to identify and authenticate every user. When people move jobs it's easy to overlook the need to remove permissions they no longer need.

Over time this can undermine carefully planned 'separation of duties' controls as users may find they retain sufficient privileges to act as both parties in a scenario which should require two individuals to collaborate.

5. All users with administrative access to your service are known. Strong authentication and access control is in place for them.

Depending on the service management and hosting models for your service, a large number of people could be in a position to access your bulk data stores, whilst avoiding some of the controls which are in place for normal users.

Privileged access may entail logical access to the service, for example through the configuration of operating systems and deployed software packages. But, it could also mean physical access to infrastructure.

Individuals with this degree of access need to be uniquely identified and authenticated with a high degree of confidence.

6. All external dependencies (e.g. third-party contractors) which the security of your service and data relies upon are known. Suppliers are vetted against defined security requirements, supported by contractual arrangements.

Suppliers who operate a service, or part of a service, are often targeted by attackers because of their access privileges. You should understand the implications of your suppliers' access being compromised and manage this effectively through your contract with them.

It is important that you explain your security requirements to your suppliers in a meaningful way, *as well as* ensuring that their contractual obligations reflect your requirements.

7. You have an audit trail of access to data, supplemented by an active audit plan.

An accurate record of which individuals accessed which data records at a given time should be captured to support an effective audit function.

It should not be possible for anyone to modify the audit record in a bid to hide the fact that they have accessed certain records. Automated alerts should be raised in response to sensitive events, such as bulk export of data or suspicious queries (such as a user accessing a record where there is no related case working ticket).

8. No known vulnerable surfaces are exposed at the edges of your service. Vulnerabilities in third-party software are mitigated. Custom software - such as web applications - is subject to testing for common vulnerabilities before handling live data. Continuous testing confirms that all of this remains true.

The vulnerabilities which an attacker can reach first are those exposed by the external interfaces of your service. Typically, these external components would be web applications, but you'll need to consider what they are in relation to your own data set or system.

Unmitigated vulnerabilities in web servers, web frameworks, third-party libraries and custom code are often easy to find and compromise using widely available, low-skill tools. If these components have access to your data set, that data is likely to be compromised when the components are exploited.

Commodity software components are easily tested for well-known vulnerabilities using vulnerability scanning tools. Custom software - such as your web applications - can also be tested. Using widely available tools it's easy to check for common vulnerabilities such as SQL injection, cross-site scripting, and cross-site request forgery.

9. No unsupported software is present in your service and its underlying infrastructure.

Software that is no longer supported will not receive security patches in the event that vulnerabilities become known. This means it will likely be difficult, or impossible, to mitigate any issues that are found.

We recommend that no 'out of support' software be used across your entire software stack for the components protecting the data. This recommendation applies to operating systems, infrastructure firmware and software packages on devices that handle or protect the data in question.

10. Basic attacks against your service would be noticed through proactive monitoring and handled through a measurable, tested, incident response process. Assume that some attacks would be successful and ensure you would detect them.

Basic attacks that could be launched by relatively unskilled attackers using widely available tools should be detected by the team operating the service, categorised according to severity, and responded to through a well-known and well-drilled process.

The sort of attacks we would consider 'basic' are:

- attempted DDoS attacks
- attempts to brute force user, service or administrator credentials
- attempts to insert malicious content into a text input field in a web form (SQL injection, cross-site scripting, cross-site request forgery)

It is good practice to test your own services for weaknesses using widely available tools and to find out if the team monitoring or operating the service detected your activities.

Plans should be in place to react to attacks against your service. System operators need to know what action to take, what they *are* authorised to do, and what decisions they would need to escalate.

11. An automatic alert would be raised in response to an atypical attempt to access bulk data. These attempts should include unusual queries, attempted large scale exports of data and administrator access to data.

Having a good understanding of what ‘normal’ looks like for your service will help you detect the abnormal.

Depending on a user’s level of authorisation and usual behaviour, queries to export large amounts of data might be entirely above board. For others this behaviour may be abnormal, and should therefore be noticed *and acted upon*.

Since administrators should have no need to access data, any administrator doing so should cause an alarm to be triggered.

12. All interfaces to your service are well-defined. None allows for arbitrary queries of the data.

If there is no function which enables users to retrieve large data sets, such functionality cannot be abused in the event of a compromise.

When building systems there’s often a desire to incorporate functions allowing users with the appropriate skills to handcraft queries and run them against the data set.

These interfaces are often prone to errors in implementation, allowing actions that were not intended. They also undermine any layered controls present in the design of the system, as they act as a bypass for the layered approach.

If there is a legitimate business requirement for an individual to have the ability to run arbitrary queries, this should be considered a privileged role, only enabled in conjunction with access logging and audit of the user’s activity.

13. User access to bulk data held by the service is rate-limited.

To avoid abuse of user access to data, or to limit the impact of a compromised account, it is advisable to apply reasonable constraints to the number of records a user is able to access, within a given period.

For example, in a case-working scenario you could limit the number of records a user can access in a shift period to a number marginally higher than the most efficient worker could manage to review within a shift period.

Similarly, if a function exists to search data sets by certain fields, then the search results should be limited to a reasonable number, based on normal expected use.

14. A spear-phishing attack against an administrator's email account, or an attack through their web browser, will not yield administrative access to the service using a single exploit.

The goal here is to mitigate the most common attack vectors used against system administrators.

If administrators use the same device or account to open their email, browse the internet, *and* administer the service, a successful attack against them would give the attacker administrative access to the service.

Some high-profile compromises of commercial services were carried out by sending a targeted spear-phishing email to a system administrator, identified through their social media profile. Poor hygiene by that system administrator is what the attackers were relying on in order to take control of the system.

‘Watering-hole attacks’ to target system administrators through sites they are likely visit should also be mitigated.

15. All backups or copies of your data are held securely, for the minimum time necessary.

Some of the most high-profile compromises of recent times stole copies (rather than primary versions) of the data from internet-connected services.

We recommend you don’t hold backups of data on internet-facing services. It is good practice to archive old records into offline encrypted backups rather than keep them within the online system. You should delete these as soon as possible.

Penetration testing by third parties can be used to help validate the security of your service, but we advise against using it as the sole means of validation.

Penetration tests only validate that you are not vulnerable to known issues *on the day of the test*. And it is not uncommon for a year or more to elapse between penetration tests. If this were your only means of validation, you could be oblivious to vulnerabilities for long periods of time.

To avoid this, we recommend you establish operational management practices which keep you well-informed of vulnerabilities present in your systems. This includes subscribing to vulnerability notification feeds on the technologies you use, and by performing your own internal vulnerability scanning and testing.

You should know what the penetration testers are going to find, before they find it. Armed with a good understanding of the vulnerabilities present in your system, you can use third-party tests to verify your own expectations. Highly skilled penetration testers can then focus on finding more subtle issues within your system.