# Security architecture anti-patterns

Six design patterns to avoid when designing computer systems.

## Introduction

At the NCSC, our technical experts provide consultancy to help SMEs and larger organisations build secure networks and systems.

This security paper describes some common patterns we often see in system designs that you should **avoid**. We'll unpick the thinking behind them, explain why the patterns are bad, and most importantly, propose better alternatives.

This paper is aimed at network designers, technical architects and security architects with responsibility for designing systems within large organisations. Technical staff within smaller organisations may also find the content useful.

↓ Download this security paper (PDF)

---

## Terminology

A few quick points on terminology before we start.

### Anti-patterns

The term 'anti-pattern' is now used to refer to any repeated (but ineffective) solution to a common problem, it is credited to Andrew Koenig who coined it in response to the seminal book 'Design Patterns: Elements of Reusable Object-Oriented Software'.

### Trust

Computer systems rarely exist in isolation. That is, they connect to networks and other systems. You might trust some of these other networks and systems more than others, and the owners of those might not trust yours at all. We use the terms:

- **less trusted** (or **low side**) to refer to the system in which we have less confidence in its integrity
- **more trusted** (or **high side**) to refer to the system in which we have more confidence in its integrity

#### Information technology vs operational technology

When thinking about trust and integrity, we consider **administration** of information technology to have broadly similar requirements to the **operation** of operational technology. Our examples below focus on the more typical information technology examples, but we think many of the concepts can be used in operational technology environments too.

---

## Anti-pattern 1: 'Browse-up' for administration

#### When administration of a system is performed from a device which is less trusted than the system being administered.

Unfortunately it is all too common to see 'browse-up' approaches to administering systems, which proves that common practice isn't always good practice. In such scenarios, an end user device used by an administrator can be one of the easiest paths into the target system, even if access is via a 'bastion host' or 'jump box'.

In computer systems where integrity is important (whether in digital services which handle personal data or payments, through to industrial control systems), if you don't have confidence in **devices** that have been used to administer or operate a system, you can't have confidence in the integrity of that **system**.

There's a common misconception that a bastion host or jump box is a good way of injecting trust into the situation, to somehow get confidence in the actions an

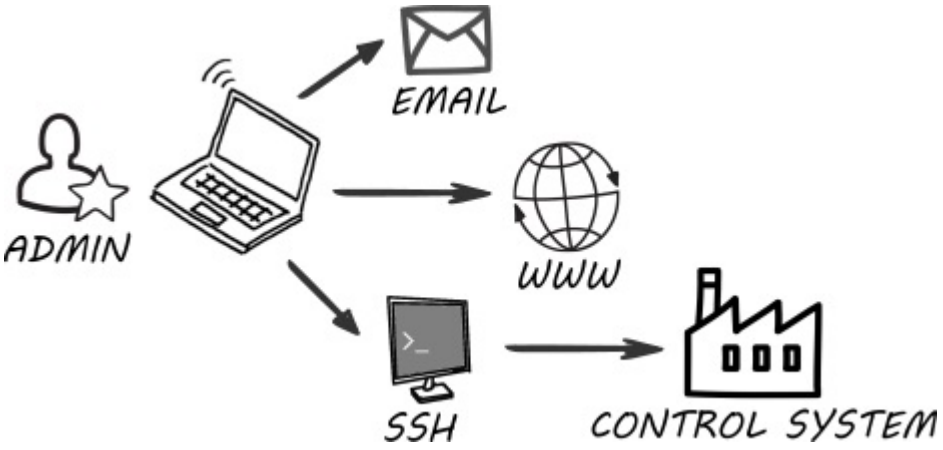
administrator is taking from a device you don't trust. Unfortunately, that's not possible.

Bastion hosts are useful for helping monitor and analyse the actions that administrators are performing, and they can help you avoid exposing more than one protocol outside of your system for administration purposes. But they won't help you be confident that the user on the device is the person you intended to allow access to. Behind the scenes, the credentials used to authenticate to the jump box could have been compromised (a reasonable assumption, given the device is less trusted). Even if administrators are authenticating their sessions with two factors, there is still the potential for malware to perform session hijacking on remote desktop or shell connections in the same way that online banking sessions are hijacked. Having gained access, the attacker can perform additional actions on behalf of the administrator. The system is under their control.

#### How to identify this anti-pattern

Here are three ways you can identify browse-up administration:

1. By looking for administration activities performed via a remote desktop (or remote shell) from a device which is part of a less trusted system.
2. By looking for outsourcing or remote support connections where a third party uses a remote desktop or shell to reach into a network. If you've got confidence in the integrity of the device used by the third party, then this isn't a browse-up problem, but if you have less confidence in their system than in yours, then it is.
3. Finally, any device which browses the web or reads external email is untrusted. So if you find an administrator using a remote desktop or shell to perform administration from the same processing context that they browse the web (or read their external email) from, then that's browsing-up too.




### A better approach: 'browse-down'

You should always use devices that you have confidence in the integrity of for administration of production systems. Those devices need to be kept hygienic (that is, they should not natively browse the web or open external email, as those are dangerous things for an administration device to do).

If, for convenience, you want to do those things from the same device, then we recommend that you 'browse-down' to do so. In a 'browse-down' model, the riskier activities are performed in a separate processing context. The strength of separation can be tailored to your needs, but the goal is to ensure that if an activity in the less trusted environment led to a compromise, then the attacker would not have any administrative access to the more trusted environment.

There are many ways in which you can build a browse-down approach. You could use a virtual machine on the administrative device to perform any activities on less trusted systems. Or you could browse-down to a remote machine over a remote desktop or shell protocol. The idea is that if the dirty (less trusted) environment gets compromised, then it's not 'underneath' the clean environment in the processing stack, and the malware operator would have their work cut out to get access to your clean environment.

### Further reading

- Microsoft Guidance: Privileged Access Management
- NCSC Guidance: Systems administration architectures

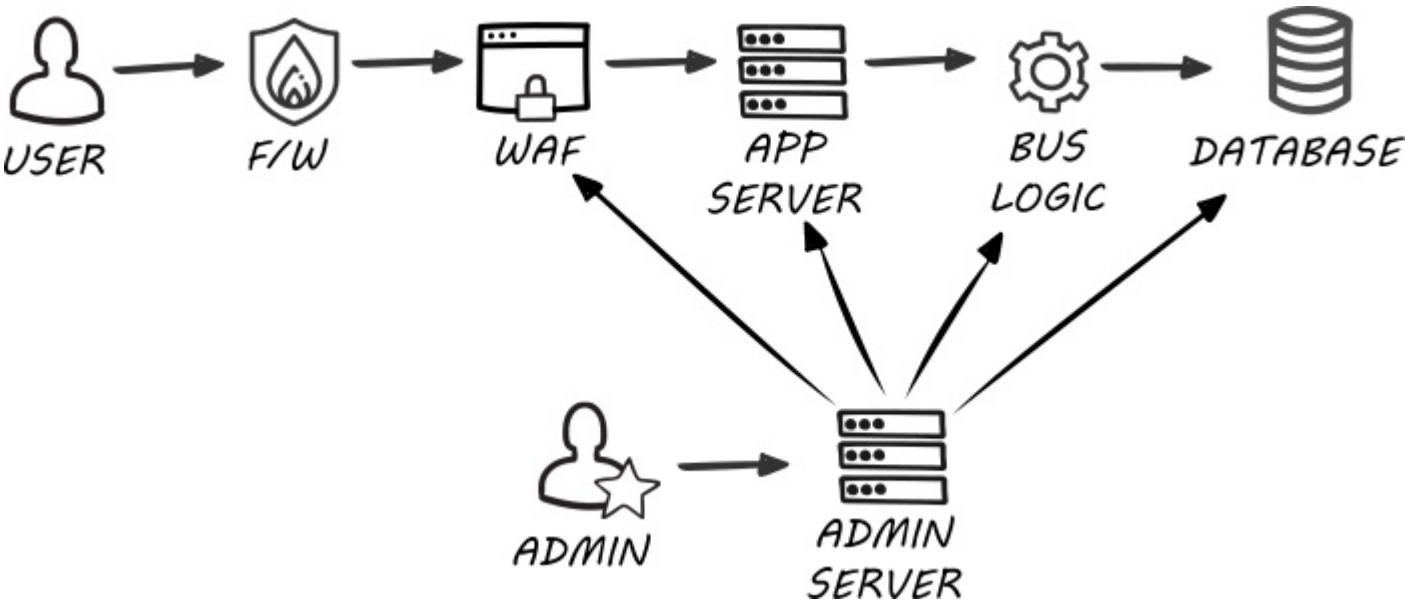
# Anti-pattern 2: Management bypass

**When layered defences in a network data plane can be short-cut via the management plane.**

It's good practice to separate management communications from the normal data or user communications on a network. In some system architectures, this would be known as separating the data plane from the management plane. However, whilst it is common to separate these types of communications with network controls, it is a common mistake to only apply the defence-in-depth concept to the data plane. If the management plane offers an easier route to the 'crown jewels' of a computer system than the data plane, then this a management bypass.

**How to identify this anti-pattern**

Look for any management interfaces from components within different layers of a system, all connected to a single switch used for management, without the corresponding layers.




**A better approach: layered defences in management planes**

The solution is simple build similar layered defences into management planes to those you have in data planes. Good practices include:

- manage from a higher trusted device, browsing down to lower trust layers
- separate bastion hosts to manage systems in each trust boundary
- different credentials for different layers to help prevent lateral movement
- restrict how systems on the data plane communicate with management plane infrastructure and vice-versa

**Further reading**

- NCSC Blog: Protect your management interfaces (contains other tips on administrative access to systems)

---

## Anti-pattern 3: Back-to-back firewalls

**When the same controls are implemented by two firewalls in series, sometimes from different manufacturers.**
There seems to be a widely believed myth that the security benefit of 'doubling up' on firewalls to implement the same set of controls is a worthwhile thing to do. Some also believe that it is preferable for the two firewalls to come from different manufacturers, their thinking being that a vulnerability in one is unlikely to be present in the other. In our experience this almost always adds additional cost, complexity, and maintenance overheads for little or no benefit.

Let's explore why we see little benefit in back-to-back firewalls in almost all cases. Take the example of an OSI layer 3/4 firewall. It has a simple job to do; control which network communications can pass through the device, and which ones can't. Putting two layer 3/4 firewalls in series is analogous to draining boiled potatoes with two colanders rather than one – it just creates more washing up.

But what if there was a vulnerability that can be exploited in a single firewall? Well, yes, that's possible. There are vulnerabilities in most things after all. But firewalls don't tend to have vulnerabilities that can be exploited to yield code

execution from processing the header of a TCP/IP packet. They tend to have vulnerabilities in their management interfaces, so you shouldn't expose their management interfaces to untrusted networks .

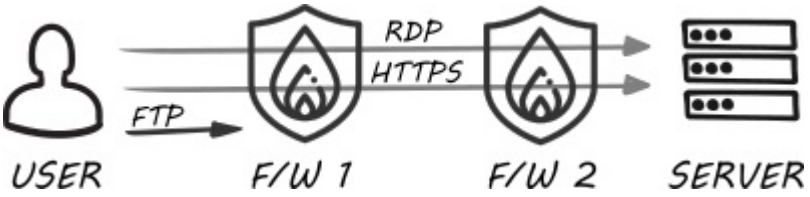
Even if there were vulnerabilities discovered in the data plane interfaces of a firewall, applying patches swiftly after their release would mean that any attack would need to exploit a zero-day vulnerability, rather than a well-known vulnerability. Furthermore, defence-in-depth design would mean that it should take more than a firewall breach to compromise sensitive data or the integrity of a critical system, and needing two zero-days to be exploited puts the attacker's capability level well beyond the threat model for most systems.

Having two firewalls would also double your admin overhead, and if you require two different vendors then you need to retain expertise in both, which adds still more cost and complexity. Plus, you have more infrastructure to maintain, and most of us find it hard enough to keep up with patching one set of network infrastructure.

However, there is one exception where we've found two firewalls to be useful; for supporting a contractual interface between two different parties. We cover this exception at the end of this section.

#### How to identify this anti-pattern

Look for two firewalls in series in a network architecture diagram.




#### A better approach: do it once, and do it well

One well-maintained, well-configured firewall or network appliance is better than two poorly maintained ones. We also recommend the following good practices:

- avoid exposing the management interfaces of network appliances to untrusted networks, and properly manage the credentials used with them

- have a simple policy configuration to reduce the chance of mistakes being introduced
- use configuration management tools to ensure you know what the configuration should be, so you can tell when it isn't correct (a tell-tale sign of compromise or internal change procedures not being followed)

#### The one exception

There is one example of using two firewalls back-to-back that makes more sense; to act as a contract enforcement point between two entities that are connecting to each other. If both parties agree on which subnets in their respective networks can communicate using which protocols, then both can ensure this is enforced by applying the agreed controls on a firewall they each manage.

#### Further reading

- NCSC Blog: Protect your management interfaces

---

## Anti-pattern 4: Building an 'on-prem' solution in the cloud

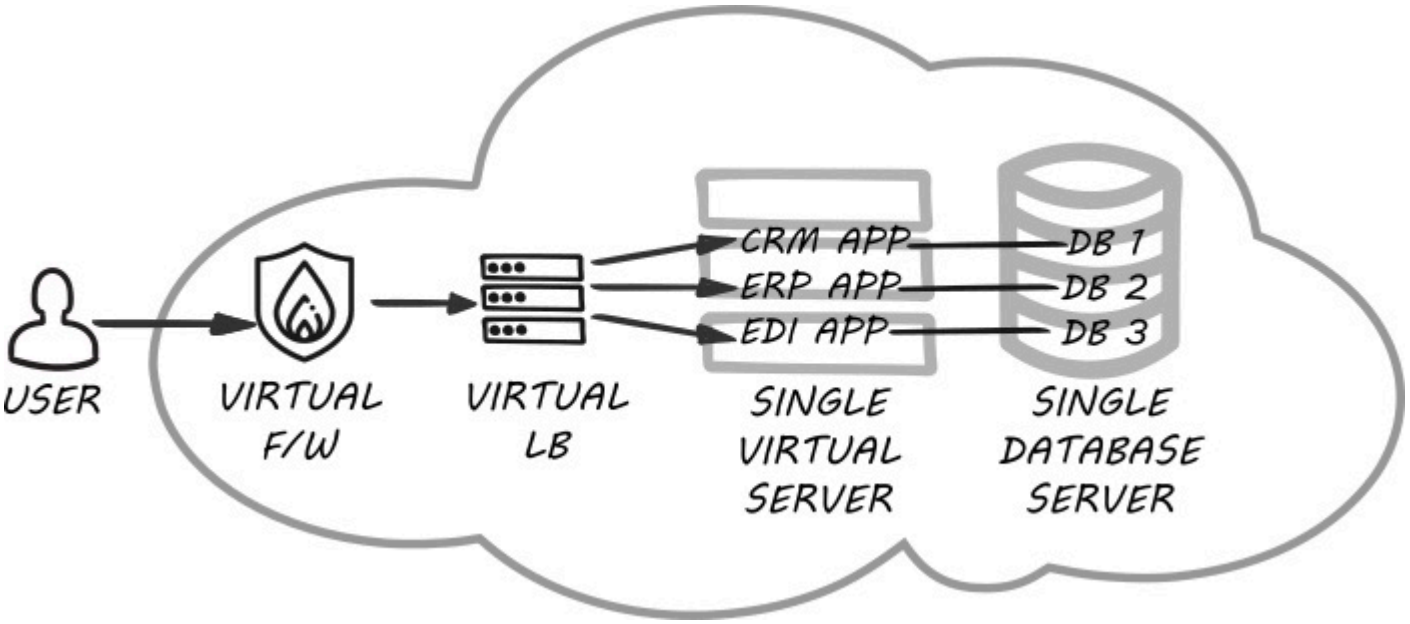
#### When you build - in the public cloud - the solution you would have built in your own data centres.

Organisations taking their first step into the public cloud often make the mistake of building the same thing they would have built within their own premises, but on top of Infrastructure-as-a-Service foundations in the public cloud. The problem with this approach is that you will retain most of the same issues you had within your on-prem infrastructure. In particular, you retain significant maintenance overheads for patching operating systems and software packages, and you probably don't benefit from the auto-scaling features that you were hoping you'd gain in the cloud.

#### How to identify this anti-pattern

Look for:

- database engines, file stores, load balancers and security appliances installed on compute instances
- separate development (and test, reference, production etc.) environments left running 24/7
- virtual appliances used without considering whether cloud-native controls would be suitable




**A better approach: use higher order functions**

Unless you're quicker at testing and deploying operating system patches than your public cloud provider is, you are probably better off letting them focus on doing that. Compare their track record of patching operating systems against your own, and judge for yourself.

Similarly, when it comes to patching database engines (or other storage services), their higher abstraction Platform-as-a-Service offerings are likely to be maintained to a level that many large enterprises will be envious of. Using higher level services like these means:

- unnecessary infrastructure management overhead is reduced
- you can focus on the things that are unique to your organisation
- your system is easier to keep patched to address known security issues

Further reading

- NCSC Blog: Debunking cloud security myths

---

## Anti-pattern 5: Uncontrolled and unobserved third party access

**When a third party has unfettered remote access for administrative or operational purposes, without any constraints or monitoring in place.**
Many organisations outsource support for some or all of their systems to a third party. This isn't necessarily a bad thing, unless done without understanding and managing the risks involved. If you outsource administration or operational functions, you're dependent on another organisation to keep your system secure. The staff, the processes and the technology of the third party all need to be considered.

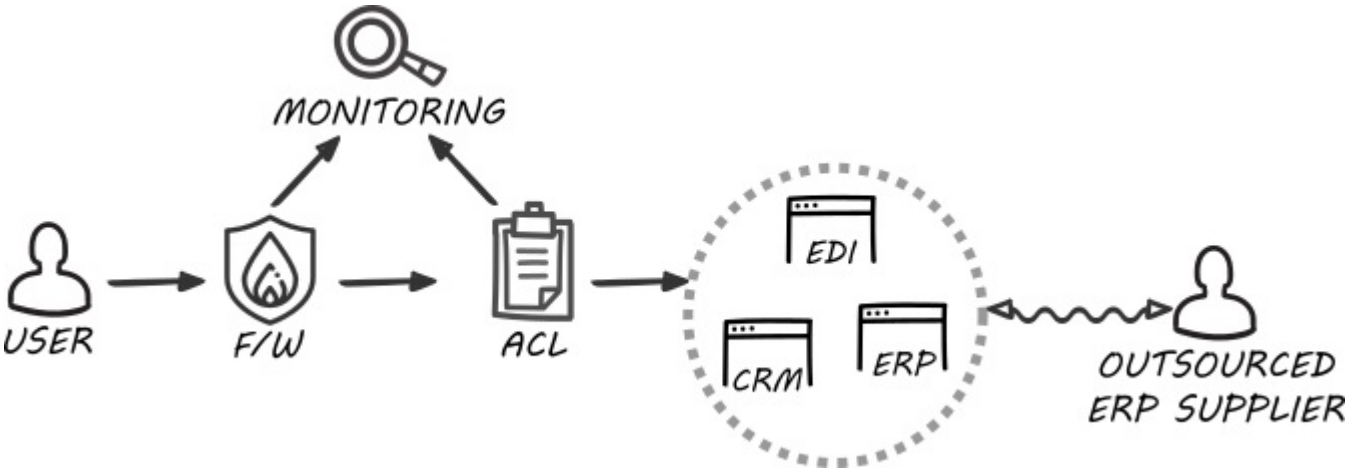
Leaving the staff and processes to one side for the moment, if a third party is administering your system, they will require access, often remotely. It's common to allow third parties to have access through a bastion host, either over the internet from allow list locations, or over a private network. However, there are often not enough controls in place to limit the operations that can be performed via the bastion host. If this is the case, and a bastion host (or the device used by the third party) is compromised, then an attacker could gain significant access to connected systems.

Let's take an example. Suppose you have purchased some niche technology that comes with a specialist support contract where the vendor needs remote access to support the device. In this case, the support organisation only needs access to the component they are supporting, and not to any other parts of your system. If you provided a bastion host that gave access to an internal network (and relied on **their** processes to only access the component they supported, rather than technically enforcing that process), then a breach of the supplier's system (or of your bastion) host would be much more damaging than it could have been.

By locking these accesses down and efficiently auditing the connection, the risk of third party compromise can be greatly reduced.

#### How to identify this anti-pattern

It's often possible to identify these relationships with third parties by looking for 'umbilical cords' out of network diagrams.




#### A better approach: a good contract, constrained access and a thorough audit trail

A good approach includes the following:

1. Choose third parties carefully with a sensible contract that sets out the controls relating to the people, processes and technology you need to have confidence in.
2. Constrained access following the principle of least privilege; only allow remote and authenticated users to have logical access to the systems they need to reach.
3. Ensure you have the audit trail you need to support incident response and support effective protective monitoring. When it comes to incident response, will you be able to confidently know which commands were executed by which user from the third-party supplier?

We also recommend the following good practices when designing a remote access solution for third parties:

- ask your supplier how they prevent attackers moving laterally between their other clients and your systems

- ensure that remote support staff use multi-factor authentication and ensure they do not share credentials - this will help you account for individual actions in event of a breach
- provide separate isolated third party access systems for different third parties
- consider using a just-in-time administration approach, only enabling remote administrative access in relation to a support ticket that is being actively worked on

#### Further reading

- NCSC Guidance: Assessing supply chain security

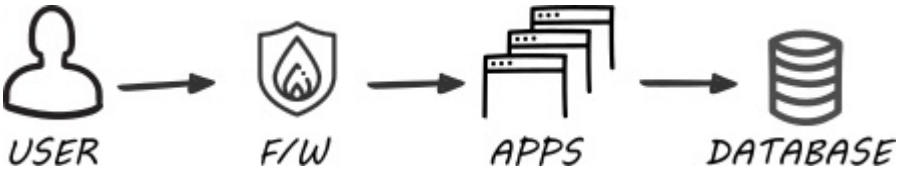
---

## Anti-pattern 6: The un-patchable system

**When a system cannot be patched due to it needing to remain operational 24/7.**
Some systems need to run 24/7. A lack of foresight could mean a system can't have security patches applied without scheduling a large window of downtime. Depending on the technologies and the complexity, it may require a window of hours (or days) to apply a patch, which could be unpalatable length of operational downtime. As time goes by, the option to defer applying security patches could mean you're left with huge number to apply during a maintenance window. Applying so many patches has now become too much of a risk, so you're trapped in a vicious circle with a system that's virtually un-patchable.

#### How to identify this anti-pattern

Look for a lack of redundancy within system architectures. Systems which require all components to be operational at all times do not lend themselves to phased upgrades, where the system could remain operational whilst undergoing maintenance.




The lack of a representative development or reference system (or ability to quickly create one) can signify a related problem. If the system owners have no confidence that the development or reference system is similar to the production system, then this can contribute to a fear of affecting stability by patching.

### A better approach: design for 'easy' maintenance, little and often

One of the NCSC's design principles is to design for easy maintenance. In some systems, this could mean ensuring you can patch a system in phases, without needing to disrupt operations. Whist this would likely require higher infrastructure cost, some of the overall lifetime costs could be lower when factoring in:

- fewer, shorter outages
- reduced risk of compromise (which could incur a costly incident response)

### Further reading

- NCSC Guidance: Vulnerability Management
- NCSC Blog: Time to KRACK the security patches out again

**PUBLISHED**

22 May 2019

**VERSION**

1.0

**WRITTEN FOR**

Large organisations

Public sector

Cyber security professionals