Features **8x Best Practice** measures for a **Vulnerability Lifecycle Management** programme

VULNERABILITY LIFECYCLE MANAGEMENT.

S4 Applications Ltd.

Mat Ludlam
https://s4applications.uk/

S4Applications

Introduction

Based upon extensive work with our large enterprise customers we have compiled a list of **eight best practice steps** to follow, for an effective vulnerability management programme.
Using market-leading technologies that protect your business, S4 Applications takes a ‘security-by-design’ approach for vulnerability assessment and remediation solutions.

Overview 8x best practice steps:

1. Build repository for data
2. Populate data on regular basis
3. Manage automatic mitigation rules
4. Understand risk
5. Generate tickets for remediation
6. Close tickets once remediation is confirmed
7. Manage risk of items that cannot be remediated
8. Report on everything

BEST PRACTICE STEPS #1 - 8 FOR AN EFFECTIVE VULNERABILITY MANAGEMENT PROGRAMME:

STEP 1
Build a data repository

You'll want to hold information on every asset connected to the network, such as the following:

- Servers Workstations
- Networking equipment
- That Samsung fridge that the intern was playing with and put on the network
- Possibly employees mobile devices depending on how you manage them
- In short everything on your network

Also you'll want information about virtual assets too, think S3 buckets and such like, along with their security profiles.

Add to that containers, and the scanned machine images that you use on AWS / Azure.
For a large company collecting over 1 Million assets would not be that difficult to imagine.

You also have to consider that it is typical to have many hundreds of pieces of information per asset.

You'll want all of this data to be interrelated in a database that can handle a capacity of 100M + pieces of data. So it would probably not be a relational database.

So next steps would be:Design a database schema that can hold all of those different types of data and allow navigation between them.

If you think about an application, you will want to see everything about it in one place, DAST and SAST results, infrastructure on the underlying servers, security issues on the S3 buckets; everything, in one place.

Implement a database structure that is capable of storing large amounts of data, and providing high speed access.

For more on how to assess your security posture, read our blog: Assess your security Posture with our Security Maturity Model.

STEP 2
Populate data on a regular basis

There are a number of different data source types that you can collect data from, including:

- Vulnerability scanners e.g. Tenable or Qualys
- Web application scanners (DAST tools) e.g. Netsparker, Acunetix, AppSpider
- Source code scanners (SAST tools), e.g. VersacodeCMDBs e.g. Service Now
- Container scanners e.g. TwistLock, AquaTicket systems (ITSMs) e.g. Service Now, Jira, HP's Service Manager
- General asset information including software inventories e.g. Microsoft's AD, Miscrosoft's SCCM, Red Hat Satellite
- SQL database CSV and XML files

Most companies have at least one of each of the items listed above.You'll need to connect securely to the source, and then extract the data.

Make sure to regularly check and maintain the secure connection, as the vendor may revise their API along with new releases.

Generating new dataSome data is inferred and would not be available in a system for import.

A typical example would be a machine name reference; consider a machine called "S-LON-PH-35".

This could mean:
- S - it is a server, not W for workstation
- LON - it is in the London data center
- PH - it is part of the Phoenix application
- 35 - a sequential number to make it unique

This data needs to be extracted from the machine name and stored for later used.

When someone says "Give me all of the machines in the London data centre" you now have that information.

Other common inferred information would be things like, if the machine is internet facing or not i.e. does it have an RFC 1918 address?

STEP 3
Manage automatic mitigation rules

Companies often have mitigation technologies that are under utilised.

Consider the situation where you have networking equipment that can perform packet analysis and block certain attacks.

WannaCry / EternalBlue is the simplest example of a particular cyber-attack. This networking equipment usually has to be enabled to instigate a check for a particular CVS. Since a search slows a unit performance down, it makes sense to have the unit look for a specific attack to accelerate the process.

Now consider the following workflow:

1. One or more machines on the network are scanned and identified as susceptible to say EternalBlue.
2. You'll need to create a ticket for the network guys to enable EternalBlue blocking on the network equipment - this is typically very quick and low risk to do.
3. You then manage remediation of the vulnerable devices on the network.
4. When every device on the network has been verified as remediated you create another ticket for the network equipment team to remove the EternalBlue block.

In the above workflow, you are vulnerable to EternalBlue for the shortest possible time frame and are maximizing your investment in the networking technology.

A similar workflow can be considered where say a SQL injection issue is found within an application. Here you would create a ticket so that a WAF rule is generated.

Again, once the core issue is resolved, the WAF rule would be removed.You always want to remove unnecessary rules as they only complicate matters, and complication is the enemy of security.

They can ultimately often slow an operation down.To learn more about vulnerability management, read out blog: Staying safe with Risk Based Vulnerability Management.

STEP 4
Understanding risk

So, now that you have everything in one place, and have metadata information about applications and such like, you can start to assign risk based on compiling several inputs.

The risk of any given vulnerability is a mixture of:

Technical properties of the vulnerability, typically given by the CVSS score and other related CVE metadata:
- Ease of exploitation?
- Is local access required?

Properties of the machine on that the CVE exists on:
Is it Internet facing?

Properties of the application that it forms part of:
- Is there personally identifiable information (PII)?
- Is there credit card information (PCI-DSS)?

Information provided by your threat intel provider?
- Is there an exploit under development?
- Is there sample code available on GitHub?

You can collate all the above information, drop it into an algorithm and score each vulnerability uniquely on a particular host.

There is no point in 90% of your vulnerabilities being 10 out of 10, or conversely 1 out of 10.

So, plan to have a wide distribution of scores to help set your priorities.

Now that you have a scoring algorithm, you need to score each vulnerability on each asset.

You can then built an asset level risk score, then rolling everything up, you can get an application level risk score.

STEP 5
Generating tickets for remediation

Now that you have all vulnerabilities and other date prioritised, you need to get the most important ones fixed / remediated.

You typically do this by creating tickets in your ITSM or ITSMs. First point to note is that most organisations have more than one ITSM, consequently different types of vulnerabilities go into different ITSMs.

Infrastructure issues may go into ServiceNow, whereas application issues (say a Cross Site Scripting issue) would probably go into a tool used by the developers; for example Jira.You also don't want one ticket per vulnerability, because that is not the way that the patching teams work.

Particularly in the Windows environment, one patch may resolve many issues, so you really only want to create tickets for Windows machines based on resolution.

The way that you create tickets, and who you assign them to will vary by platform and a number of other variables.

If you consider, say a Cisco router, running an old version of Cisco's IOS, this will have a long list of vulnerabilities.

You can upgrade it to the current version and most of those vulnerabilities will be removed with a single upgrade.

This is very different to a programmer who has 3 SQL Injection issues within a single application, they may just prefer 1 ticket and deal with all 3 issues at once.

STEP 6
Close tickets once remediation is confirmed

Once the remediation teams have applied the fix, they will update their ticket. You could trust that the work is complete and close the ticket down.

The best plan though is to trigger a re-scan of the associated item, and have the scanning technology confirm remediation.

When the scanning technology confirms remediation, then the ticket can be closed.Note that this can be harder than it sounds because many of the scanning technologies don't report items as remediated, they just stop reporting their existence.

This means you have to compare the outputs from different runs. You also have to be careful, because not all runs are the same.

If last week you ran a full scan and this week a quick scan, then they cannot be compared.Vulnerability X, that was found last week, was not looked for in the quick scan.

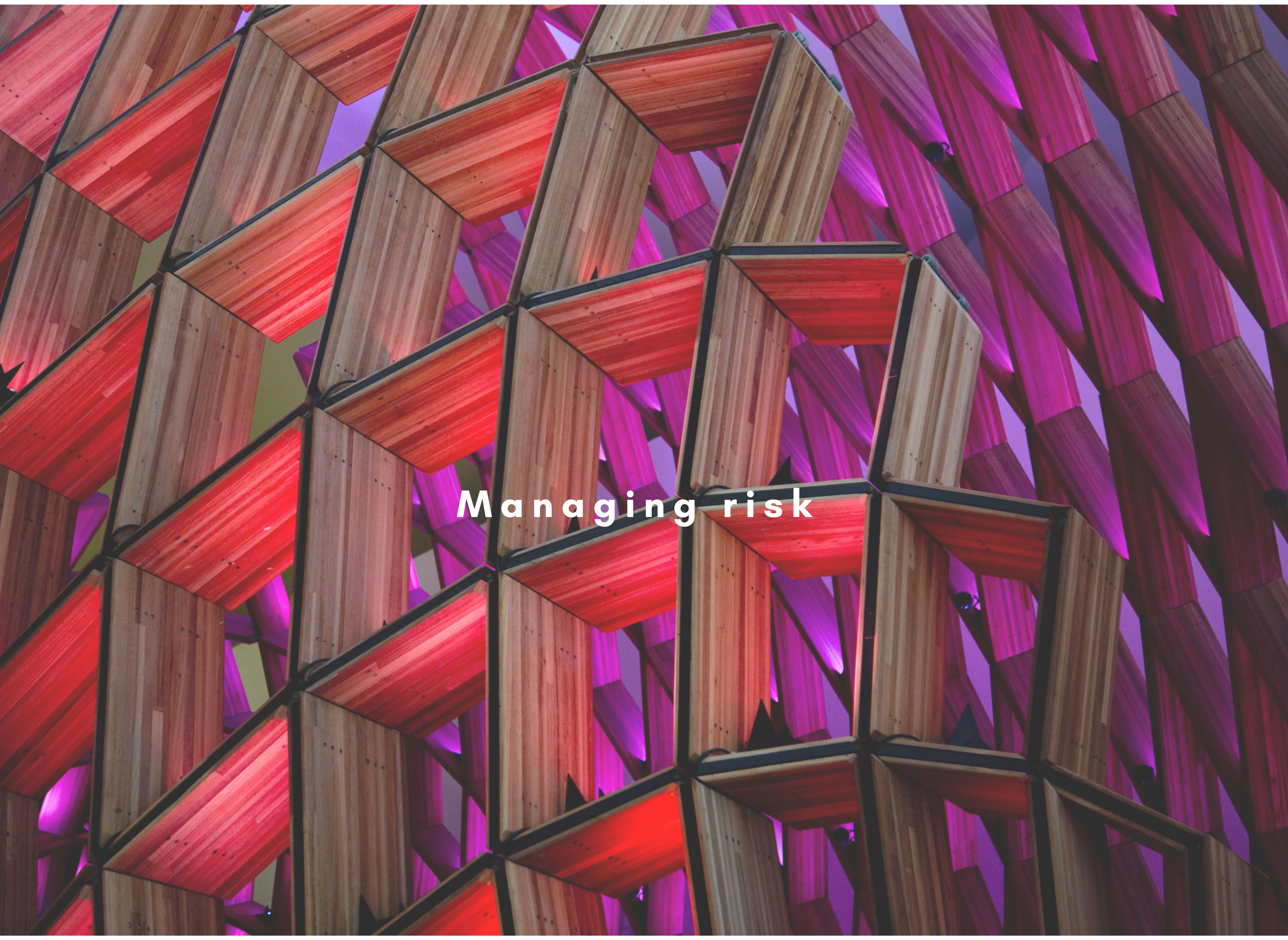
This doesn't mean that vulnerability X has been fixed, only that it was not checked for.In the event that the vulnerability still exists, then the ticket is pushed back to the remediation team.

This is often because a re-boot has yet to be completed or similar!!

This also has the benefit of closing out vulnerabilities that were remediated as a by-product of the remediation work.

So consider a Windows Patch Tuesday update. There will probably be at least one critical item that you have to patch. In the process of applying that patch you will probably fix some other, less important issues at the same time.

To learn more about the difference between Vulnerability Scanning and Pen testing read our blog.



STEP 7
Managing risk of those items that cannot be remediated

The most common example of this is Java but impacts a lot of other technologies too.

Consider an example where the vulnerability scanner says that the version of Java in use must be upgraded because it is full of holes.

The issue is that the hosted application is not supported on the newer Java version, so it cannot be upgraded.

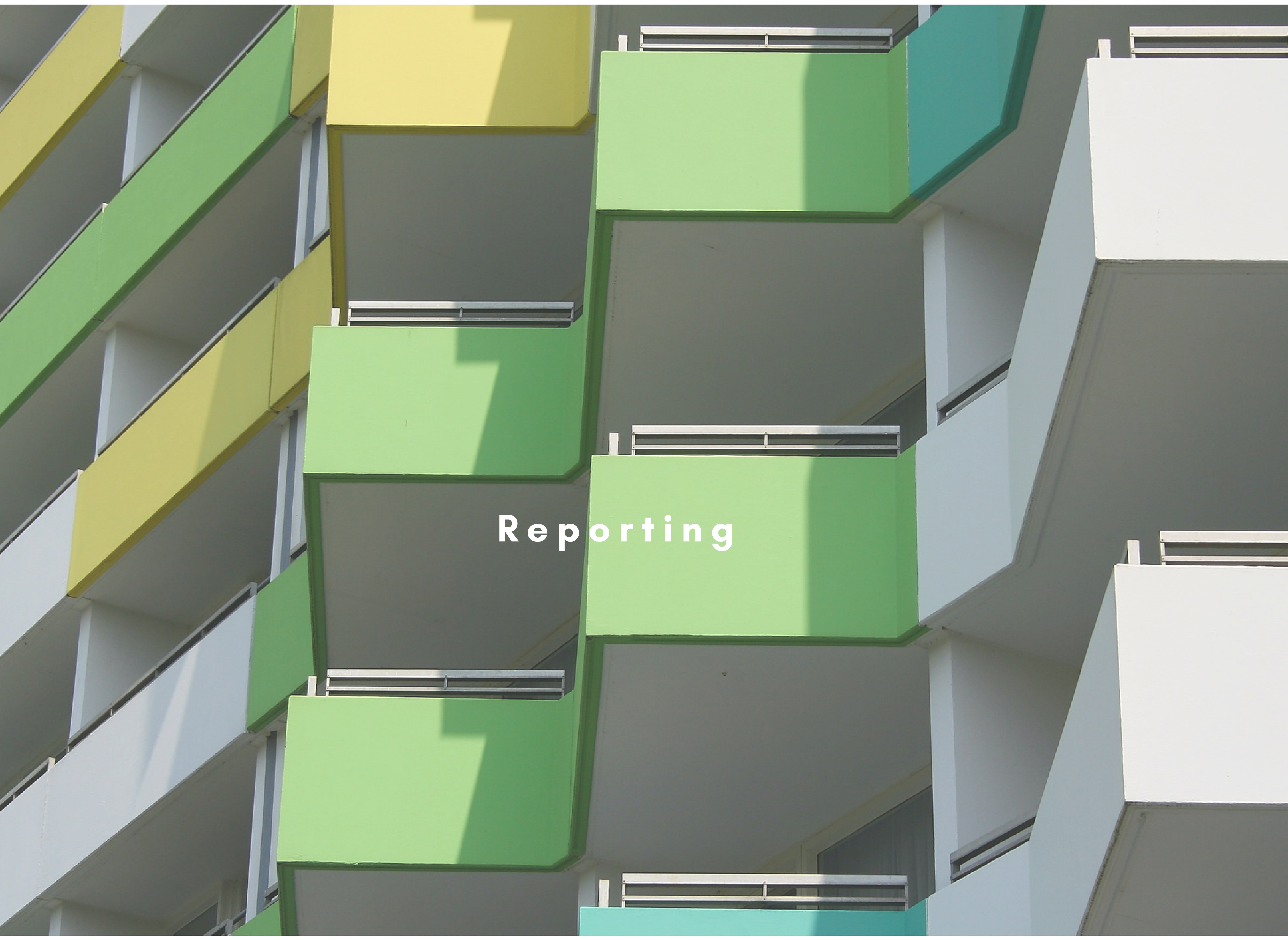
Here you need a risk management system. The remediation team flag an item as being a risk.

There then needs to be some form of approval process where specific mitigations are discussed and any residual risk reviewed by the business.

If the business decides to accept the risk, then it should only be for a limited amount of time, and the risk will need to be reviewed and accepted again.

The process then becomes something like:

- Remediation team identifies something that cannot be remediated.
- Case file is built and discussed, including:
  - Current risk
  - Possible mitigations and their effectiveness
  - Replacement time and risk during this period
  - Actions to be taken
- Business decides if residual risk can be accepted.
- Actions agreed are completed.
- The accepted riskis reviewed on a given timeframe. The review timeframe varies
- based upon risk, with higher risk items being reviewed more often.
- When review occurs then start back at stage 2 because many assumptions will have changed.



STEP 8
Report on everything

Lots and lots of data, means:Everything needs to be reported upon.

Create dashboards with different needs for different stakeholders.
Drill through the data.
Measure tickets against agreed SLAs.
See on one screen an application, assess all of the host servers.
Review all the vulnerabilities on those servers.

All of the DAST and SAST findings in that application.

Data security - people from London data centre shouldn’t see the Singapore data.

Next steps:

Brinqa vulnerability management solutions.

Brinqa prioritises assets, vulnerabilities, and incidents based on their impact and value to your business.

Read more about risk based vulnerability management with our Brinqa case studies, learn more about Brinqa or contact S4 Applications to request a demo.

s4 Applications Ltd.
sales@s4applications.uk