OpenSecOps SOAR

Standard Operating Procedure

Table of Contents

Document Versions

Version	Date	Changes	Author
1.0	2024-02-14	First version	Peter Bengtson
1.1	2025-04-07	Replaced “Delegat” with “OpenSecOps” throughout	Peter Bengtson

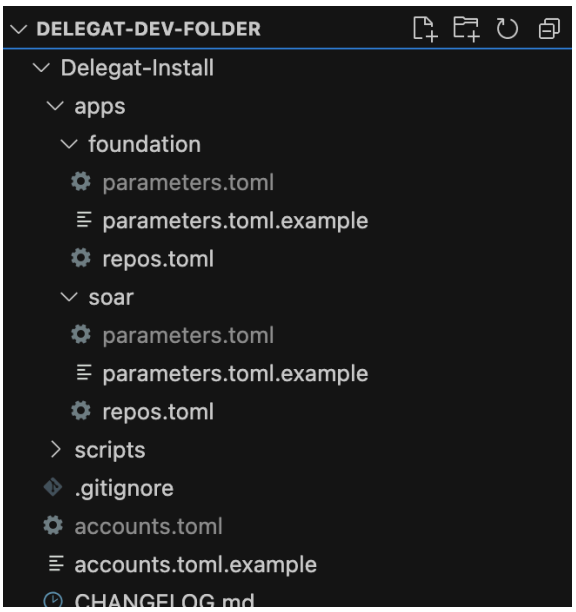
1 Introduction

The purpose of this document is to specify how OpenSecOps SOAR is configured and maintained, including organisational procedures to follow, so that infrastructural operations personnel and security architects can gain a full understanding of how to configure and operate it.

2 Configuration Parameters

Initial Configuration

Below is a list of OpenSecOps SOAR configuration parameters to configure at installation. All parameter configuration is done in the OpenSecOps Installer repo. You can find the configuration files here:



Please refer to the README of the OpenSecOps Installer repo for information of how to create and set up the grayed-out configuration files from the example files given in the repo. All you have to do is copy the example files to create the real files, then edit them appropriately.

First set up `accounts.toml`, as this file lets the installer know your account IDs and SSO profile names. Then edit `apps/soar/parameters.toml` to configure OpenSecOps SOAR specifically. You will find the most important configuration parameters at the top of the file, or marked "`Configure this`" further down in the file. Most of the parameters can be left to their default values.

The list of configuration parameters below includes only the ones you need to configure for a typical installation. Should you, for instance, want to customise the tag names for accounts or the paths for secrets this can be done too. The configuration file exposes all parameters available to you.

NB: `accounts.toml` and `apps/soar/parameters.toml` are excluded from source control. Any changes to them will remain on your local computer. You will want to back them up to a safe company-dependent location.

Updating the SOAR Configuration

Changing any configuration parameter requires a redeploy of OpenSecOps SOAR, which is done using OpenSecOps Installer.

For configuration not requiring a redeploy, see Chapter 4 on the DynamoDB tables.

Global Parameters

The following parameters are used across all repos part of OpenSecOps SOAR.

org-id
: Your AWS Organizations ID. It has the form "`o-xxxxxxxxxx`".

root-ou
: The ID of your AWS Organization's ROOT OU. It has the form "`r-xxxx`".

main-region
: Your main AWS region into which the SOAR will be installed, such as "`eu-north-1`".

other-regions
: A list of regions apart from the `main-region` in which OpenSecOps SOAR is to operate, e.g., `["us-east-1"]`, or `["eu-west-1", "us-east-1"]`. If you only want to monitor `main-region`, provide the empty list: `[]`.

cross-account-role
: This role enables the system to operate across different organisational accounts. Depending on your AWS setup, specify one of the following:

    - `AWSControlTowerExecution` if you are under AWS Control Tower
    - `OrganizationAccountAccessRole` if you are under AWS Organizations without Control Tower.

email-domain
: The email domain to use for notifications, e.g. "`example.com`", "`acme.cloud`", etc. This domain must already exist, and AWS SES must be able to send to it. It is easiest to set up a special domain for the purpose on AWS. Do not forget to enable DKIM and take the domain out of the sandbox and into production status.

SOAR General Settings

DiskForensicsInvokeARN

Blank by default.
If the SAM version of the Disk Forensics Collection project is installed, enter the ARN of its invocation lambda which can be found as the output `DiskForensicsInvokeARN` in the Security account. If this parameter is left

blank, rogue EC2 instances will simply be terminated. If the value of DiskForensicsInvokeARN is entered here, the Disk Forensics Collection project will snapshot and terminate them instead. Ticketing and emailing remains the same.

MinAgeHours

The default value is `24`.
The number of hours of respite a newly created account is given to stabilise before OpenSecOps SOAR starts processing events for the account.

ClearAccountDataCacheRate

By default, this is "`6 hours`".
How often the account data cache is cleared. You will want to set this to a value equal to or smaller than MinAgeHours. NB: the string must be grammatically correct. "`1 hour`", "`2 hours`", "`24 hours`", etc. (Incorrect values will signal an error during deployment.)

SOAR Email Settings

SendEmail

The default setting is `'Yes'`.
Choose whether to send emails. Selecting `'Yes'` will enable email notifications, while `'No'` will disable them.

Allowed values:

- `Yes`
- `No`

EmailCC

The default CC address is `'soc@example.com'`.
Any email sent will have a carbon copy (CC) sent to the addresses listed here. You can specify multiple addresses by separating them with commas. Update this value based on the team or individuals you want to keep in the loop. You can safely set EmailCC to the empty string.

EmailBCC

Left blank by default.
This parameter allows you to specify blind carbon copy (BCC) addresses. Recipients listed here will receive the emails, but other recipients won't see them. List multiple addresses by separating them with commas.

DefaultTeamEmail

By default, this is left blank.
Provide an email address here if you want a default address to be used when specific team emails aren't given by an account's tags. If this field is left empty, the system will default to using the main account email.

SOAR Ticketing Settings

IgnoreProducts

Default: `'Inspector,Systems Manager Patch Manager'`.

Security Hub products listed here will be ignored during incident creation. Separate products with commas. You might want to consider disabling ingestion of ASFF events instead of suppressing them in this way. AWS Inspector might generate hundreds or even thousands of incidents after a vulnerability scan, and you might not want to create an incident for each one of them.

IncidentsToSoc

Whether incidents should also generate tickets for SOC. Allowed values:

- `NONE` – No, they shouldn't.
- `INFRA` – Yes, but only the infrastructural ones (GuardDuty, etc).
- `APP` – Yes, but only for application-generated incidents.
- `ALL` – Yes, all of them should.

The default value is `INFRA`.

SocJiraProjectKeyOrServiceNowQueue

For issuing tickets to SOC. If using JIRA, a JIRA project key. If using ServiceNow, a queue name. Set to blank to suppress SOC ticketing.
Default: `'SOC'`

TicketingSystem

Defaults to `'JIRA'`.
Choose the ticketing system you're using. This can be `JIRA`, `ServiceNow`, or you can select `'None'` if you don't want to integrate with any ticketing system. This might be useful during initial setup and tuning.

Allowed values:

- `JIRA`
- `ServiceNow`
- `None`

JIRA Settings

You only need to specify this if you are using JIRA for ticketing.

JiraDefaultProjectKey

Default project key is `'XXX'`.
This is the default abbreviation for the Jira project. It's used when a specific JiraProjectKeyTag isn't provided.

ServiceNow Settings

You only need to specify these if you are using ServiceNow for ticketing.

ServiceNowTable

By default, it uses the `'incidents'` table.
Specify which ServiceNow table to use for ticket creation.

ServiceNowDefaultProjectQueue

Default queue is `'XXX'`.
The default queue in ServiceNow. Used when a specific ServiceNowProjectQueueTag isn't provided.

SOAR Overdue Ticket Settings

For optimal security posture and efficient operational handling, it's important to keep track of ticket lifetimes. Depending on the severity of a ticket, different response times are expected. This section defines the time limits, in hours, that tickets of various severities can remain open before they are considered overdue:

SeverityAllowedAgeInHoursCritical

Default: `4`
Tickets with a CRITICAL severity should ideally be resolved within 4 hours. If a CRITICAL ticket is open for longer than this duration, it's considered overdue.
Special Case: If set to -1, CRITICAL severity tickets will never be marked as overdue.

SeverityAllowedAgeInHoursHigh

Default: `8`
Tickets with a HIGH severity should be addressed within 8 hours. Any HIGH ticket open for more than this duration is classified as overdue.
Special Case: If set to -1, HIGH severity tickets will never be considered overdue.

SeverityAllowedAgeInHoursMedium

Default: `336` (=14 days)
Tickets with a MEDIUM severity have a more lenient response time. They should be resolved within 336 hours or 14 days. Beyond this time frame, the ticket will be flagged as overdue.
Special Case: If set to -1, MEDIUM severity tickets will never be considered overdue.

SeverityAllowedAgeInHoursLow

Default: `672` (=28 days)
Tickets with a LOW severity have the most flexible response time. They should be addressed within 672 hours or 28 days. If a LOW ticket exceeds this duration, it becomes overdue.
Special Case: If set to -1, LOW severity tickets will never be marked as overdue.

It's crucial to periodically review and address open tickets to maintain security standards and operational efficiency. Adjust the above limits as needed based on organisational needs and operational capacity.

SOAR AI Settings

This section outlines the configurations for the AI capabilities, particularly those leveraging OpenAI's ChatGPT. These configurations empower the system with AI capabilities while ensuring security and data privacy. Adjust the settings as needed based on organisational requirements and data policies.

Using ChatGPT

UseChatGPT

Default: `No`
Determines if the system should use OpenAI's API for ChatGPT. If set to `Yes`, you must have installed the GPT API key at `/soar/chatgpt/api-key` in Parameter Store.
Allowed Values: `Yes`, `No`

ChatGPTIaCSnippets

Default: "`Cloudformation YAML, Terraform, and Python CDK`".
This configuration lets you define the Infrastructure as Code (IaC) languages in which you wish your teams to get infrastructural snippet suggestions. Note that the above is part of a sentence given to GPT, so keep it grammatically correct for best results. Examples:

- "`Cloudformation YAML and TypeScript CDK`"
- "`Terraform`"
- "`Python CDK and Cloudformation JSON`"

Weekly AI-Based Security Report

WeeklyReport

Default: `No`
If set to "`Yes`", a weekly AI-driven security report will be sent to the designated recipients. If set to `Yes`, you must have installed the GPT API key at `/soar/chatgpt/api-key` in Parameter Store.
Allowed Values: Yes, No

WeeklyReportEmailRecipients

Default: ""
Defines the recipient(s) of the full weekly AI-based security report.

WeeklyReportIndividualAccounts

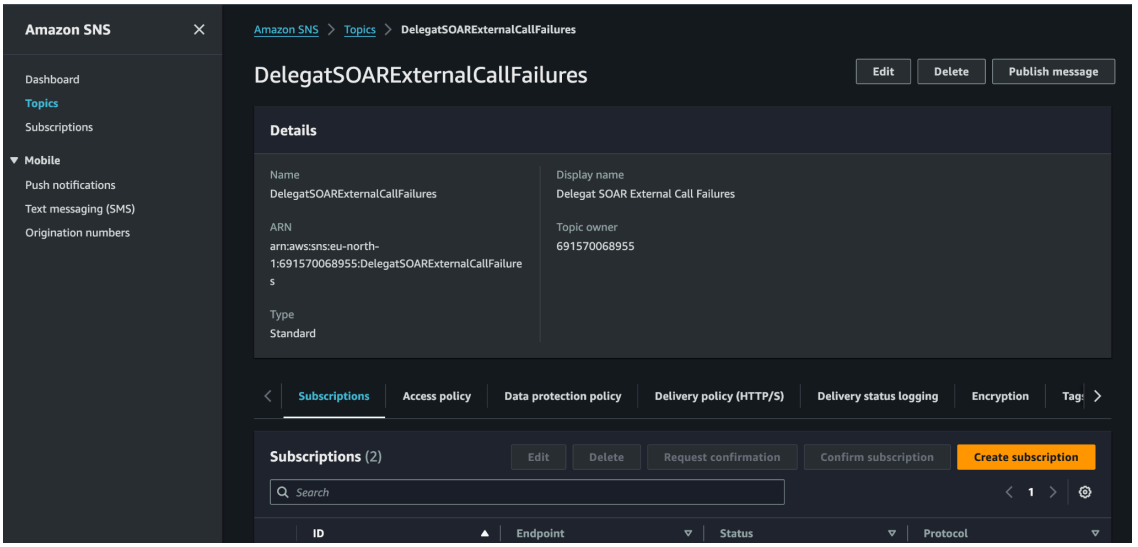
Default: `Yes`
If set to "`Yes`", individual account owners will be sent the sections of the weekly full report pertaining to the accounts they manage.
Allowed Values: `Yes`, `No`

3 External Calls SNS Error Topic

If desired, OpenSecOps SOAR can integrate with external web services for ticketing (Jira, ServiceNow), generative AI (the OpenAI ChatGPT API), and incident management (Microsoft Sentinel).

As OpenSecOps SOAR is the hub for all security-related processing, it must not be affected by the availability of such external web services. Errors trying to reach external APIs in order to process security issues must not give rise to further security issues.

For this reason, all failures trying to call to services external to AWS are instead reported through an Amazon Simple Notification Service (SNS) topic, `OpenSecOpsSOARExternalCallFailures`. You can find the topic in the main organisational account, in your main region:

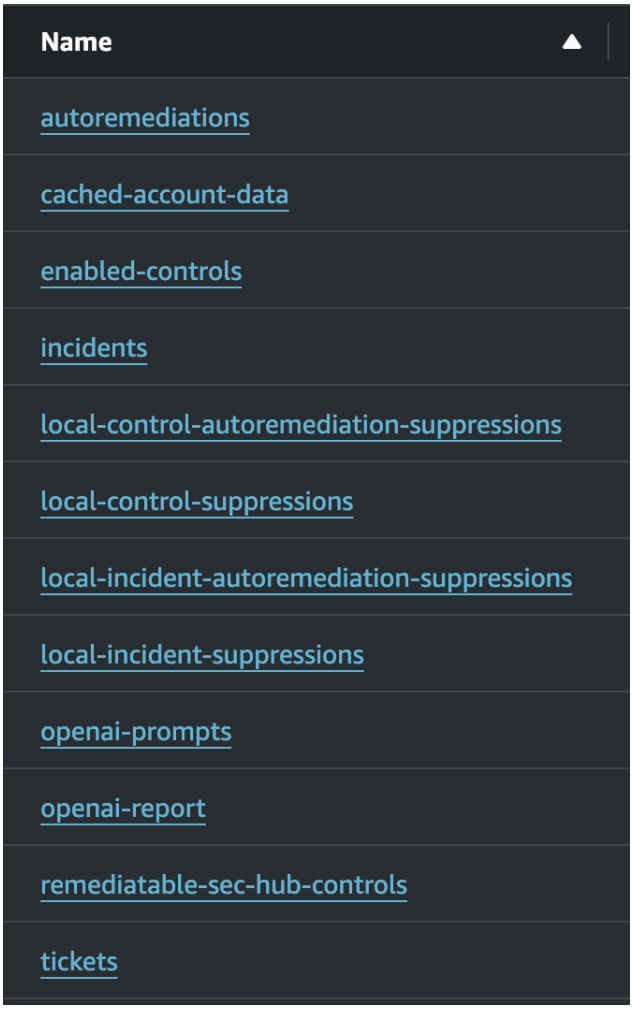


We recommend you to subscribe to the topic, so you can receive email notifications should external calls fail.

It should be added that when an external call fails, the SOAR will adapt accordingly. If a call to the OpenAI API fails for a new ticket, for instance, the email to the team involved will use the standard information in the ASFF issue instead of the AI-enhanced versions, thus degrading in a controlled fashion.

4 The DynamoDB Tables

OpenSecOps SOAR creates twelve DynamoDB tables in your main region:



The main purpose of these tables is to allow further configuration, to cache account data, and to keep track of control issues, autoremediations performed, and incidents received and processed.

Tickets, Autoremediations, Incidents

The tables "`autoremediations`", "`incidents`", and "`tickets`" contain items for each ticket, autoremediation, and incident generated during the past year. The "`tickets`" table is used to keep track of open and/or overdue control failures. All three of these are used for statistical purposes and report generation. None of them should be touched by the user.

Caches and Prompts

The tables "`cached-account-data`", "`enabled-controls`" and "`openai-report`" serve as caches. They are fully managed by the SOAR and do not need to be

touched by the user. Similarly, "`openai-prompts`" contains GPT prompts used by the AI layer and is entirely managed by the OpenSecOps Installer.

Enabled Autoremediations

This table, "`remediatable-sec-hub-controls`", contains the IDs of the Security Hub controls for which we want to enable auto-remediation. It is set up automatically as part of the installation of OpenSecOps SOAR.

Initially, all autoremediations are enabled, but their execution is dependent on the corresponding control. For instance, if you have disabled the control that detects whether an ALB redirects from HTTP to HTTPS, its corresponding autoremediation won't be performed.

For this reason, there's very little point in editing the contents of "`remediatable-sec-hub-controls`", although it certainly is possible. One situation where you'd want to do this is when you want the issue to be detected but always result in a ticket to the team, as if the autoremediation for the issue didn't exist.

Each item consists only of the ID of the Security Hub control. Simply delete the ones you do not require.

If you do not want to modify this table, an alternative method is to disable the autoremediation using a suppression rule in one of the tables described in the next section.

Local Suppression of Controls, Incidents, and Autoremediations

The remaining four tables are used to suppress issues in your infrastructure. Usually this is done locally, e.g., for a specific environment or OU, or for specific accounts or teams, but it can also be globally, for every account.

An example of this might be the rule for not deleting KMS keys: it is important never to delete a KMS key in `PROD` as doing so might result in permanent loss of business data, but you might in all likelihood want to permit it in your development and test environments. Similarly, you might wish to enforce multi-AZ database deployment in production only. You can achieve this by using Security Hub's centralised configuration of controls, of course, but you can also achieve the same result using the suppression tables.

The four tables controlling this are:

- `local-control-suppressions`
- `local-control-autoremediation-suppressions`
- `local-incident-suppressions`
- `local-incident-autoremediation-suppressions`

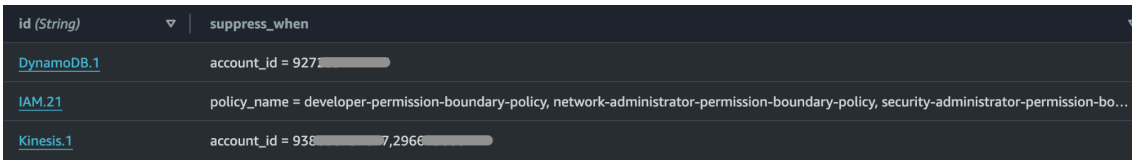
As you can see, suppression also extends to any autoremediations available.

Each table has the same structure: each item has only two properties:

- "`id`" – the control or incident ID,
- "`suppress_when`" – when to disable the control, incident, or autoremediation.

Before we proceed to how to create `suppress_when` expressions, let's take a look at what is set up automatically during installation for each table:

local-control-suppressions



DynamoDB.1, which requires all DynamoDB tables to have capacity elasticity, is relaxed in the AFT-Management account.

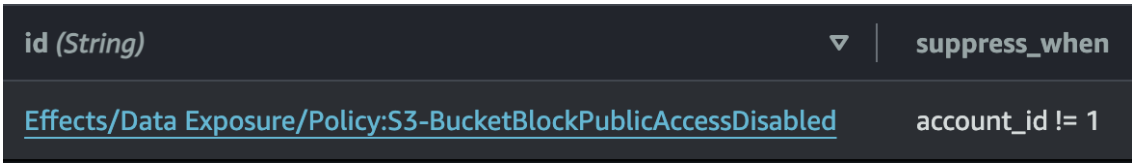
IAM.21, which prohibits user-managed policies from using wildcards for services (such as "`s3:*`") is relaxed for OpenSecOps Foundation's built-in permission boundary policies for Developers, Network administrators, and Security administrators.

Similarly with Kinesis.1 (Kinesis streams should be KMS-encrypted at rest) which we relax in Log Archive and the administrative AWS Organizations account.

local-control-autoremediation-suppressions

This table is initially empty. You can add entries to it to inhibit autoremediations locally for specific accounts, OUs, environments, etc.

local-incident-suppressions



The above disables the GuardDuty S3 incident signalled when an S3 bucket is discovered to have its Block Public Access setting disabled. The condition will always be true, which means that the incident will be suppressed globally. The reason this is disabled this way is because the incident is already handled by the machinery we've built around opening S3 buckets for public read and/or write access, and thus is a false positive.

If, on the other hand, your use cases involves prohibiting all external access to S3 buckets, you should remove this suppression rule from the table.

local-incident-autoremediation-suppressions

There is only one auto-remediation for incidents, the GuardDuty EC2 `MEDIUM`, `HIGH` and `CRITICAL` incidents to which we respond by simply terminating the instance. There are quite a few GuardDuty EC2 incidents of this severity.

This table is currently empty.

Local Suppression Rule DSL

This rule DSL is used to specify whether a specific Security Hub control or incident is to be suppressed locally, i.e. in one or more parts of the infrastructure.

- "`id`" must always be present
- "`suppress_when`" must always be present. If not present, or if the value is the empty string, the control or incident will not be suppressed.

Condition Dimensions

The string in `suppress_when` can specify a set of conditions which match the finding's account properties. If they match, the finding will be suppressed.

You can create conditions built on a finding account's:

- **account_id** - the account in which the issue was created (12 digits)
- **region** - the region in which the issue was created
- **client** - the client with which that account is associated
- **environment** - the environment of that account
- **organizational_unit** - the OU to which the account belongs
- **project_name** - the name of the project to which the account belongs
- **team** - the team/squad/group working in the account

And also on the following resource property:

- **policy_name** - if the involved resource is an IAM Policy, its name

Equality and Member of List

The syntax for a line is best described by examples:

```
account_id = 111111111111
```

This means "suppress the control's findings in account 111111111111".

You can also specify a list on the right-hand side of the expression:

```
organizational_unit = ROOT, INFRA, SANDBOXES
```

The above means, "suppress the control's findings in organisational units `ROOT`, `INFRA`, and `SANDBOXES`. The equal sign thus means, "true if the value is in the list".

Inequality and Not Member of List

You can also specify that an account should *not* match a specific condition:

```
environment != PROD
```

The above means, "suppress the control's findings everywhere but in the `PROD` environment".

Similarly, you can test for non-membership of a list:

```
environment != DEV, TEST
```

The above demonstrates non-equality with lists. It means, "suppress the control's findings in all environments except in DEV and TEST. Negation thus means, "true if the value is NOT in the list".

AND

It is also possible to combine conditions using AND:

```
environment = DEV AND team = Platform
```

which means, "suppress the control's findings in the DEV environment of the Platform team".

Note that `AND` must be in uppercase and on the same line as both expressions.

OR

The value of `suppress_when` can be multiline. If so, the lines form an implicit logical OR. If any line evaluates to True, then the issue will be suppressed. If all lines evaluate to False, the issue will not be suppressed.

Example:

```
environment = DEV
organizational_unit = INFRA AND account_id != 222222222222
```

The above means, "suppress the control's findings in all DEV environments and also throughout the entire OU INFRA except in account 222222222222".

Note that the actual token `OR` is never used. A multi-line value for `suppress_when` will implicitly be interpreted as a logical OR.

5 New Security Hub Controls

As part of the ongoing development of Security Hub, AWS releases new Security Hub controls at irregular intervals. Sometime after the introduction of a new AWS service, there might appear new controls to help monitor the new service. Alternatively, additional controls for existing services might be added to encourage or enforce best cloud practices at a deeper level.

In the default setup of AWS Security Hub, all such new controls are enabled by default. This is not desirable in an enterprise setting, especially not one that creates tickets with SLAs of the silver bullet kind, as this would mean that AWS in practice would directly dictate what developers and operations personnel do. Also, new controls do not appear simultaneously in all regions but are rolled out over time. Frequently, the first rollout takes place in North Virginia (us-east-1).

Instead, what is needed is a process to determine which new controls should be activated. For instance, AWS once introduced a new control that mandated the creation of a Dead Letter Queue for every lambda. This is impractical, especially since this also would be required of all third-party software. Also, AWS retracted the control after some time.

Thus, OpenSecOps SOAR disables the automatic enabling of new controls during installation. All new controls must be enabled manually by an administrator logging into the Security account and enabling them in the Security Hub console in each region.

However, before this can be done, there is a process to follow:

1. As a security engineer, regularly check each region for new controls. You can always see them on the News tab in Security Hub.

2. When there are enough controls to proceed (one might suffice), create an overview of the new controls, what each one does, their severities, what consequences this might have for existing infrastructure, and when you think it should be enabled and why.

3. Run this list by your architects and security bodies to verify your conclusions.

4. When you have a definite list of new controls to enable, decide on a date or a number of dates for enabling them. You might wish to do this very gradually or all at once depending on their nature.

5. Communicate the timetable to all stakeholders.

6. Enable the new controls according to the timetable, notifying all parties each time something is enabled.