ONE-PERSON IT OPERATIONS HANDBOOK

# ひとり情シス大全

ひとり情シスが会社を止めないための実務


版

2026年3月版

発行

haya株式会社

Contents

# 目次

## 第IV部　セキュリティ、法務、監査



## 第V部　障害、インシデント、事業継続



## 第VI部　改善、自動化、AI、引き継ぎ



## 付録と巻末

# はじめに

ひとり情シスという言葉は、肩書きより状態を表していることが多い。専任の情報システム担当であっても、総務や管理部門との兼務であっても、現場で起きていることは似ている。PC の準備、アカウント運用、SaaS 管理、障害対応、セキュリティ対策、契約更新、ベンダー調整、経営への説明まで、会社の IT に関わる実務が一人へ集まりやすい。

この本は、そうした状態に置かれた人が、場当たり対応だけで毎日を終わらせないための本である。単なるツール紹介や気合の話ではなく、会社を止めないために何を知り、何を決め、どう回すかを、できるだけ体系立てて整理した。

本書で重視したのは、ひとり情シスが必要になりうる知識をできるだけ漏らさないこと と 実務で使える順序に並べること である。そのため、日常運用だけでなく、調達、契約、外部委託、個人情報保護、障害対応、ランサムウェア、BCP、生成 AI ガバナンス、引き継ぎまで含めた。忙しい現場ほど、問題が起きてから必要な知識を探しに行く余裕がないからである。

ただし、すべてを一度に整える必要はない。本書は 全部やるべきこと一覧 として読むより、いま最も弱い場所を見つけて次の一手を決める本 として使った方が役に立つ。小さな会社では、網羅性より順序の方が大事である。

なお、製品名、管理画面名、制度運用、公式ガイドの版は時間とともに変わる。本文では、変わりにくい原則と、変わりやすい実務の線をなるべく分けて書いたが、導入や運用の直前には公式情報の再確認を前提にしてほしい。

本書が目指す到達点は、何でも知っている担当者を育てることではない。自分しか回せない状態を減らし、会社に知識と運営が残る形へ近づけることである。

# 本書の読み方

本書は、第1章から順に読んでもよいが、実務では いま困っている場所 から読む方が使いやすい。次の目安で入ると進めやすい。

## まず土台を作りたい時

- 第1章から第3章で、ひとり情シスの守備範囲、運営の土台、仕事の回し方をつかむ
- 第4章から第7章で、経営、契約、外部委託、導入変更の考え方を固める

## 日常運用を整えたい時

- 第8章から第16章で、アカウント、入退社、端末、ネットワーク、SaaS、共有、バックアップ、依頼管理を確認する
- 台帳、申請、窓口、標準手順を先に整えると、後半の章も回しやすくなる

## セキュリティや有事対応を見直したい時

- 第17章から第23章で、基礎対策、脆弱性管理、個人情報保護、教育、障害、重大事故、BCP を確認する
- 事故が起きてから読むのではなく、平時に どこまで準備できているか を点検する読み方が向いている

## 改善と次年度計画へつなげたい時

- 第24章から第27章で、自動化、可視化、生成 AI、引き継ぎと次の一手を確認する
- 忙しさを減らすには、便利なツール導入だけでなく、運用の持ち方を変える必要がある

## 読み進める時の見方

- 各章では、まず全体像をつかみ、その後に 通常時の例と、崩れた時の例 を見る
- 章末の 最低限ここまではやる は、最初の実行ラインとして使う
- 今日、今週、後でやること は、その章を読んだ直後の行動へ落とすための欄である

## この本の前提

- 小規模から中規模の組織で、ひとりまたは少人数で IT 運営を担う読者を主対象にしている
- 特定製品の完全操作マニュアルではなく、運営判断と実務の原則を扱う
- 会社ごとに規模、業種、法規制、利用製品は違うため、本文は 最小限の現実解 を基準にしている

第I部

# 全体像と運営の土台

守備範囲と仕事の型を最初に固める。

第1章〜第3章

第1章

# ひとり情シスとは何を担う仕事か

朝、出社すると、新入社員用のノート PC が届いていないことに気づく。先にアカウントだけは発行しておきたいので、クラウドの管理画面を開いてユーザーを追加する。その途中で、営業部から「商談先で Wi-Fi につながらないと困るので、今日中に会議室の無線を見てほしい」と連絡が入る。昼前には、退職者のメールを誰に引き継ぐか総務から相談が来る。午後は、月末で切れる SaaS の契約更新を確認し、夕方には取引先から届いたセキュリティチェックシートに回答する。

これが、ひとり情シスの一日として特別に誇張された光景かというと、そうではない。むしろ、少し静かな日の方に入る。

ひとり情シスの仕事は、こうした雑多な作業の寄せ集めに見えやすい。だからこそ、自分でも周囲でも、「何をやっている人なのか」が曖昧になりやすい。すると、仕事の優先順位が定まらず、引き継ぎもできず、事故が起きたときだけ存在が強く意識される状態になる。

この章の目的は、その曖昧さをなくすことにある。ひとり情シスとは何を担う仕事なのかを、最初に言葉で定義する。ここが曖昧なままだと、後の章で扱う端末管理も、アカウント管理も、セキュリティも、ただの項目集になってしまう。逆に、ここで守備範囲の地図を持てれば、自分に足りないものと、今すぐ整えるべきものが見えやすくなる。

## ひとり情シスは役職名ではなく、配置の名前である

まず押さえたいのは、ひとり情シスは厳密な職種名ではないということだ。会社によって肩書きは、総務、管理部、経営企画、店舗運営、社長室、情報システム担当などさまざまである。実際には、正式な「情報システム部」がなくても、誰か一人が IT に関する判断、調整、設定、問い合わせ、事故対応を引き受けていれば、その人は実質的にひとり情シスである。

この状態を理解するには、「ひとり情シスは一つの専門職ではなく、複数の役割が一人に集まった状態だ」と考える方が分かりやすい。本来なら、会社の IT には、問い合わせ対応をする役割、アカウントを管理する役割、端末やネットワークを運用する役割、ベンダーを調整する役割、セキュリティ事故に備える役割などがある。規模の大きい会社では、これらは複数人や複数部署に分かれる。しかし、中小企業や少人数組織では、そうした役割が分かれず、一人に集まりやすい。

2024年2月に公開された NIST Cybersecurity Framework 2.0 は、サイバーセキュリティを `Govern` `Identify` `Protect` `Detect` `Respond` `Recover` の 6 つの機能で整理した。ここで重要なのは、守るだけでなく、統治し、把握し、検知し、対応し、復旧するまでが一つの流れとして扱われていることだ。ひとり情シスの実務も同じである。端末を配るだけでも、終わりではない。どの端末を誰が使い、どの権限が与えられ、異常が起きたときにどう対応し、退職時にどう回収し、記録をどう残すかまでを考えなければならない。

つまり、ひとり情シスとは「PC に詳しい人」ではない。会社の IT サービスを、限られた人員で何とか運用し続けるために、本来分かれている複数の役割をつないでいる人のことだ。

## ひとり情シスの守備範囲は、想像よりずっと広い

ひとり情シスの守備範囲は、次の七つに大きく分けて考えると整理しやすい。

第一に、問い合わせ対応と依頼対応である。パスワード再設定、ソフト導入、アカウント発行、印刷できない、VPN につながらない、といった日々の困りごとを受ける入口になる。周囲から最も見えやすい仕事なので、ここだけがひとり情シスの仕事だと思われやすい。

第二に、アカウント、認証、権限の管理である。入社時の発行、異動時の変更、退職時の停止、MFA の適用、共有アカウントの整理、管理者権限の分離などが入る。最近の IT 環境では、端末やネットワークそのものより、まず ID と認証が入口になる。ここが弱いと、多くの対策が意味を失う。

第三に、端末、資産、ネットワークの管理である。PC　やスマートフォンのキッティング、配布、回収、交換、紛失対応、Wi-Fi　や回線、会議室機器や複合機まで、業務に使う物理的な基盤を支える。

第四に、SaaS、クラウド、データの管理である。今の会社では、業務の多くがクラウド上で動いている。メール、ファイル共有、チャット、勤怠、経費、営業支援、顧客管理などがそれぞれ別の SaaS に分かれていてもおかしくない。ひとり情シスは、それらを個別に管理するだけでなく、アカウント連携、外部共有、データの保存先、解約時のデータ返却まで見なければならない。

第五に、セキュリティ、インシデント対応、復旧である。IPA　の中小企業向けガイドラインでも、経営者向けの指針と実務担当者向けの手順の両方が必要だとされている。ウイルス対策ソフトを入れるだけで終わる話ではない。怪しいメールをどう防ぐか、管理者権限をどう分けるか、事故が起きたら誰に何を連絡するか、復旧後に何を見直すかまで含む。

第六に、予算、契約、ベンダー、導入である。現場はここを情シスの仕事と思っていないことが多い。しかし、何を買うか、どの契約で入れるか、誰に委託するか、障害時の責任分界をどうするかは、運用品質を大きく左右する。安く入れたつもりが、後から運用負荷が膨らむことは珍しくない。

第七に、規程、監査、教育、改善である。利用ルール、台帳、申請フロー、監査証跡、教育、ナレッジ整備、業務改善、自動化などがここに入る。普段は目立たないが、これがないと、すべての仕事が個人依存になる。

この七つの領域は、別々に存在しているわけではない。たとえば退職者対応を考えれば、アカウント停止だけでなく、メール引き継ぎ、端末回収、ライセンス回収、データ保全、管理者権限の棚卸し、記録保存までつながる。ひとり情シスの難しさは、一つひとつの技術が高度だからではない。別の領域だと思われている仕事が、実際には一本の業務としてつながっていることにある。

## ひとり情シスの仕事は、会社の事業継続と信用に直結する

ひとり情シスの仕事は、裏方の雑務だと軽く見られやすい。だが、実態は逆である。会社の仕事は、メール、認証、端末、ネットワーク、ファイル共有、SaaSが動くことを前提に回っている。これらのどれか一つが止まるだけで、営業、受発注、請求、顧客対応、人事手続きまで広く影響を受ける。

さらに、影響は社内だけにとどまらない。IPA　が　2025年5月27日に公開した2024年度　中小企業における情報セキュリティ対策に関する実態調査　では、中小企業の対策不足は、自社被害だけでなく、取引先への影響や攻撃の踏み台化につながると整理されている。規模が小さいから狙われない、専任情シスがいないから仕方がない、という理屈は通りにくくなっている。

また、情シスの役割は守りだけでもない。経済産業省の DX 支援ガイダンスが示すように、中堅・中小企業では人材、情報、資金の不足により DX の独力推進が難しい。だからこそ、現場の業務を知り、既存の IT 基盤も把握しているひとり情シスが、SaaS　導入や業務改善の実務上の支えになる場面が増える。つまり、ひとり情シスは「コスト部門の保守係」ではなく、守りと攻めの両方の土台に立つ存在である。

## ひとり情シスに対するよくある誤解

ここで、ひとり情シスに関する典型的な誤解を五つ整理しておきたい。

一つ目は、「PC　　に詳しければ務まる」という誤解である。確かに、端末やネットワークの基礎知識は必要だ。しかし、実務ではそれだけでは足りない。契約更新、アカウント棚卸し、退職対応、ベンダー調整、利用ルール整備のように、技術だけでは処理できない仕事が多い。

二つ目は、「セキュリティは別の話」という誤解である。実際には、アカウント発行も、端末配布も、SaaS 導入も、退職者対応も、すべてセキュリティとつながっている。後からセキュリティを追加すればよいというものではない。

三つ目は、「ベンダーに任せれば終わる」という誤解である。委託は重要だが、責任分界を決めるのは自社である。誰が最終判断をするのか、どの情報を渡すのか、障害時に誰へ連絡するのかが曖昧なら、委託は丸投げになり、事故時に最も弱い形で破綻する。

四つ目は、「忙しいから記録は後回しでよい」という誤解である。実際には逆で、忙しいからこそ記録が必要だ。台帳、手順書、申請記録、契約一覧、管理者アカウントの保管ルールがないと、同じ確認を何度も繰り返し、退職や障害のたびにゼロから調べ直すことになる。

五つ目は、「ひとり情シスは一人で全部抱えるのが正しい」という誤解である。ひとり情シスとは、仕事を一人で全部やる人のことではない。本来は分散しているべき仕事を、少人数でなんとかつないでいる状態である。だからこそ、標準化、外注、ルール化、自動化、経営への相談が必要になる。

## ひとり情シスに求められる成果は、作業量ではなく運用品質である

ひとり情シスは、やった作業の数で評価しにくい。チケットを何件処理したかよりも、事故を起こしにくい運用になっているか、止まりにくい基盤になっているか、引き継げる状態になっているかの方が重要である。

本書では、ひとり情シスに求められる成果を、次の五つで捉える。

第一に、止まらないことである。日常業務が回り続ける状態を保つ。大きな改革より、まず止めないことが優先される局面は多い。

第二に、漏れないことである。入社、異動、退職、契約更新、権限変更、バックアップ、ログ保存などで、抜け漏れが起きないようにする。

第三に、残ることである。台帳、手順、判断理由、連絡先、契約情報が記録として残り、自分が不在でも最低限追える状態を作る。

第四に、増えすぎないことである。例外運用、個別対応、野良　SaaS、属人的な設定、管理者権限の乱立を放置すると、将来の自分の首を絞める。仕事を増やさない設計も成果である。

第五に、次に改善できることである。今日の火消しだけで終わらず、同じ問題を減らし、少しずつ標準化し、自動化し、経営と共有できる状態を作る。

この五つを見れば、ひとり情シスの成果は「なんでもすぐやること」ではないと分かる。むしろ、なんでも自分で瞬間対応してしまうほど、運用は残らず、後で崩れやすくなる。

## ひとり情シスが詰まりやすい場所

ひとり情シスは、能力不足より、構造上の詰まりで苦しくなることが多い。典型的なのは次のような場面である。

問い合わせの入口がばらばらで、メール、チャット、口頭、電話に依頼が散っている。すると、優先順位がつけにくく、未対応が発生しやすい。

台帳がない、または古い。端末、アカウント、契約、管理者権限のどれか一つでも曖昧だと、退職対応や障害対応で必ず詰まる。

退職者対応が単発作業になっている。アカウント停止はしたが、共有フォルダの所有権移管やライセンス回収が漏れる、といったことが起きる。

契約更新や証明書更新の期限管理が弱い。普段は問題にならないが、切れた瞬間に事業影響が出る。

バックアップはあるが、復元確認をしていない。いざ戻そうとして初めて、手順がない、権限がない、世代が足りない、保存先が巻き込まれていた、と気づく。

管理者権限が個人依存している。前任者しか知らないアカウント、古いスマートフォンにしか入っていない認証、誰も管理していない共有メールボックスは、典型的な事故の種である。

こうした詰まりは、どれも特別な大事故ではない。日常の延長線上で起きる。そして、起きてから対処すると必ず高くつく。だから本書では、第2章以降で、土台、回し方、対象別管理、有事対応を順に扱っていく。

## 最低限ここまではやる

この章の段階では、すべてを完璧に理解する必要はない。ただし、次の四つを押さえれば十分である。

一つ目。ひとり情シスは、問い合わせ対応係ではなく、会社の IT サービス全体をつなぐ役割である。

二つ目。その仕事は、技術、統制、調整、記録、改善が混ざった複合業務である。

三つ目。業務範囲を意識的に整理しないと、事故は個人の忙しさではなく、構造の問題として起きる。

四つ目。だから、本書では「一つひとつを詳しく知る」前に、「全体をどう捉えるか」を先に整える必要がある。

**確認したいこと**

- 自社で自分が担っている IT 業務を 10 個以上書き出せる
- その業務が、問い合わせ、アカウント、端末、SaaS、セキュリティ、契約、規程のどこに属するか分類できる
- 自分の仕事を「PC の設定」ではなく「会社の IT サービス運営」と言い換えられる
- 今の自分の仕事で、個人依存しているものを 3 つ挙げられる
- 後続章のうち、まず読むべき章を自分で選べる

## 今日、今週、後でやること

今日やることは、30　分だけ取り、自分が抱えている仕事を棚卸しすることである。端末、アカウント、契約、問い合わせ、障害対応、外部委託、規程の七つに分けて書き出すだけでよい。その作業をすると、自分が何を担っていて、何が抜けているかが見え始める。

今週やることは、書き出した仕事を 問い合わせ 日常運用 導入変更 セキュリティ契約調達 有事対応 のようなまとまりで見直し、どこに個人依存が強いかを三つ選ぶことである。

後でやることは、本書を読み進めながら、自分の棚卸し結果へ章ごとの学びを追記していくことである。そうすると、この本は読むだけの本ではなく、自社の運営整理ノートとして使える。

第2章

# 最初に整えるべき運営の土台

朝からチャットで「印刷できません」と連絡が来る。数分後に口頭で「新しいメンバーのアカウント、今日中に作れますか」と聞かれる。その間に、メールで「来月更新の　SaaS、契約どうしますか」と問い合わせが届く。昨日頼まれたノート　PC　の手配は、どこまで進めたか記録がない。先週設定した共有フォルダの権限変更も、誰に何を許可したのか思い出せない。

ひとり情シスが苦しくなるのは、仕事の件数が多いからだけではない。もっと本質的な理由は、依頼が散り、記録が残らず、必要な情報がどこにあるか分からないまま、毎回その場で判断しているからである。

ここでよく起きる誤解がある。「もう少し落ち着いたら整えよう」「まずは詳しいツールを調べよう」「時間ができたら手順書を書こう」という考え方だ。しかし実際には逆で、土台がないから落ち着かず、仕組みがないから毎回時間がなくなる。

この章では、ひとり情シスが最初に整えるべき五つの土台を示す。窓口、記録、台帳、手順、優先順位である。どれも地味だが、この五つがない状態では、どんな高度なセキュリティ対策も、どんな便利な SaaS も、長くは回らない。

## まず整えるべきものは何か

運営の土台として、最初に必要なのは次の五つである。

一つ目は、窓口である。誰が、どこに、何を依頼するのかを決める。

二つ目は、記録である。何を受け、誰が持ち、どこまで進んだかを後から追えるようにする。

三つ目は、台帳である。端末、アカウント、契約、ライセンスなど、運用の対象を一覧で把握できるようにする。

四つ目は、手順である。よくある作業を毎回考え直さずに済むようにする。

五つ目は、優先順位である。声の大きさではなく、影響の大きさで仕事を並べる。

この五つは、それぞれが独立しているわけではない。窓口が一本化されると記録しやすくなる。記録が増えると、よくある依頼が見える。よくある依頼が見えると、手順が書ける。手順が書けると、優先順位に従って落ち着いて処理しやすくなる。土台とは、こうして互いに支え合うものだ。

重要なのは、最初から完璧な仕組みを作らなくてよいということである。共有メールアドレスとスプレッドシートでも始められる。最初に必要なのは、立派さではなく、一つの流れを作ることだ。

## 問い合わせ窓口を一本化する

仕事が崩れる最初の原因は、入口が多すぎることにある。メール、チャット、口頭、電話、廊下ですれ違った時の一言まで、すべてを依頼として受けていると、どれが未対応で、どれが急ぎで、どれが完了したのか分からなくなる。

だから、最初に決めるべきなのは窓口である。理想を言えば、チケットツールやポータルがあるとよい。しかし、最初の一歩としては共有メールアドレスでも十分だ。`it@company.co.jp` のような窓口を一つ作り、「IT の依頼はまずここへ送ってください」と決める。それだけでも、仕事の見え方は大きく変わる。

ここで大切なのは、窓口を作ること以上に、そこへ寄せる習慣を作ることだ。チャットで直接頼まれたとしても、「対応はするので、この窓口にも入れてください」と返す。口頭で言われたら、自分で代筆してもよい。最初から全員が守るわけではないが、窓口へ寄せる動きを続けると、少しずつ入口が揃ってくる。

窓口を一本化すると、三つの効果がある。第一に、依頼の見落としが減る。第二に、今どれだけ仕事を抱えているかが見える。第三に、後から振り返って「何に時間を使っているのか」を把握できる。これは、後の改善や外注判断にもつながる。

## 依頼を記録し、分類する

窓口を一本化したら、次は記録である。ここでいう記録とは、立派な運用台帳を作ることではない。最低限、後から追えることが大事だ。

記録には、少なくとも次の項目が必要である。

- 受付日時
- 依頼者
- 内容
- 分類
- 優先度
- 状態
- 担当
- 完了日

もしチケットツールを使うなら、これらは自然に入る。使わないなら、スプレッドシートで十分である。重要なのは、どの道具を使うかではなく、毎回同じ項目で残すことだ。

分類は、最初から細かくしなくてよい。まずは三つでよい。

- 依頼
  - アカウント発行、ソフト導入、ライセンス追加など
- 障害
  - メールが使えない、Wi-Fi が落ちた、VPN がつながらないなど
- 変更
  - 設定変更、権限設計変更、新しい運用ルール導入など

この三つを分けるだけでも、仕事の性質が見えやすくなる。依頼は定型化しやすい。障害は初動が重要である。変更は影響確認と記録が大切になる。全部を同じ箱に入れると、処理の考え方まで混ざってしまう。

ここでありがちな失敗は、「記録すること自体が面倒だから、頭の中で管理する」ことである。頭の中の管理は、忙しい間は回っているように見える。しかし、人に聞かれた時、二週間前の依頼を思い出したい時、自分が休んだ時に必ず破綻する。ひとり情シスほど、記憶力ではなく記録に頼るべきである。

## 台帳を作る

ひとり情シスの仕事で詰まりやすい場面の多くは、「何を持っているか分からない」ことから始まる。端末が何台あるのか、誰がどの PC を持っているのか、どのアカウントが生きているのか、どの SaaS を契約していて更新日はいつか。これが曖昧だと、退職対応も、契約更新も、障害対応も、棚卸しもすべて苦しくなる。

最初に必要な台帳は、四つでよい。

一つ目は端末台帳である。端末名、利用者、機種、貸与日、保管場所、状態が分かればよい。

二つ目はアカウント台帳である。氏名、部署、利用サービス、権限の種類、管理者権限の有無が見えるようにする。

三つ目は契約台帳である。サービス名、契約先、契約期間、更新日、解約条件、連絡先、支払方法を記録する。

四つ目はライセンス台帳である。契約数、使用数、未使用数、割当先、見直し時期を残す。

ここでも最初から完璧な項目を揃える必要はない。大事なのは、「何がどこに書いてあるか」が決まっていることだ。退職者が出た時に、端末台帳を見れば PC が分かる。アカウント台帳を見れば止めるべきサービスが分かる。ライセンス台帳を見れば回収できるものが分かる。契約台帳を見ればベンダー連絡先と更新条件が分かる。この状態が作れれば、仕事は急に軽くなる。

## 手順を残す

ひとり情シスの疲労は、仕事の量だけでなく、「毎回同じことを最初から考える」ことでも増える。パスワード再設定、新規 PC の準備、共有フォルダの権限付与、退職者対応、ライセンス追加。何度も発生する作業は、手順として残しておくべきである。

ここで注意したいのは、手順書を完璧なマニュアルにしようとしないことだ。最初は短くてよい。見出しと箇条書きでもよい。重要なのは、対応しながら残すことだ。

たとえば、パスワード再設定の依頼が来たら、その対応の流れを 5 行でもよいので書く。新しい PC を渡したら、キッティング時に確認した項目をチェックリストにする。退職者対応をしたら、止めたサービスや回収物を一覧化する。これを繰り返すと、頭の中にしかない知識が、少しずつ組織の知識に変わっていく。

Atlassian の KCS が示すように、ナレッジは問い合わせ対応の後にまとめて書くものではなく、対応の一部として育てるものだ。ひとり情シスにとって、この考え方は特に重要である。後で書こうとして書けるほど、仕事は軽くならないからだ。

## 優先順位を決める

依頼が増えた時、最後に物を言うのは気合いではなく、優先順位である。優先順位が決まっていないと、近くにいる人、声の大きい人、役職が上の人、先に連絡してきた人に引っ張られやすくなる。しかし、本来の基準はそこではない。

最初に必要なのは、単純でよいので、緊急度と影響度の考え方を持つことだ。

緊急度とは、どれだけ早く対処が必要かである。影響度とは、どれだけ多くの人や重要な業務に影響するかである。

たとえば、一人だけが困っているプリンタ設定の依頼と、全社の VPN 障害は同じではない。新規ソフト導入の相談と、退職者のアカウント停止も同じではない。前者は急ぎに見えても、後者の方が事業影響やリスクが大きいことがある。

最初の運用では、次のような単純な基準で十分である。

- 高
  - 全社または重要業務が止まる
  - セキュリティ事故の可能性がある
  - 退職や権限停止のように即日性が高い

- 中
  - 一部部署が困る
  - 期限が近い
  - 業務影響はあるが代替可能
- 低
  - 個別依頼
  - 後日でも支障が小さい
  - 定型作業として計画的に処理できる

大切なのは、この基準を自分の中だけに置かないことだ。紙一枚でもよいので残し、説明できる形にしておく。そうすると、依頼者に対しても「なぜ今すぐできないのか」「なぜこちらを先にやるのか」を伝えやすくなる。

## 最初から完璧を目指さない

ここまで読むと、「結局やることが多い」と感じるかもしれない。だが、第2章で伝えたいのは、完成形を一気に作れという話ではない。むしろ逆である。最初から立派な ITSM を作ろうとすると、仕組みだけが重くなり、運用が止まる。

だから、最初の三か月は次のような最小構成でよい。

- 共有窓口を一つ決める
- 依頼記録を一つの表で始める
- 端末、アカウント、契約、ライセンスの台帳を仮でも作る
- よくある作業を三つだけ手順化する
- 優先順位ルールを一枚で決める

この状態でまず回してみる。すると、不足が見えてくる。問い合わせの分類が足りない、契約台帳の項目が不足している、退職対応のチェックリストが必要だ、といった具体的な改善が見える。その段階で初めて、道具や制度を足せばよい。

ひとり情シスの運営基盤は、最初から設計し切るものではない。回しながら整えるものである。ただし、何もない状態から回しながら整えることはできない。だからこそ、最初の骨格だけは先に作る必要がある。

## 最低限ここまではやる

この章の段階では、次の五つがあれば十分に前進である。

一つ目。IT の依頼窓口が一つ決まっていること。

二つ目。依頼や障害を後から追える記録があること。

三つ目。端末、アカウント、契約、ライセンスの最低限の台帳があること。

四つ目。よくある作業の手順が少しでも残っていること。

五つ目。優先順位を説明できること。

これだけで、仕事は突然楽になるわけではない。しかし、少なくとも毎回ゼロからやり直す状態からは抜けられる。ひとり情シスに必要なのは、まずこの変化である。

**確認したいこと**

- IT の問い合わせ窓口が一つ決まっている
- 依頼を受付日と状態つきで記録している
- 端末台帳がある

- アカウント台帳がある
- 契約台帳とライセンス台帳がある
- よくある作業の手順が三つ以上ある
- 優先順位ルールを言葉で説明できる

## 今日、今週、後でやること

今日やることは多くない。まず、共有窓口を一つ決める。次に、スプレッドシートでよいので、依頼記録と四つの台帳を作る。最後に、よくある作業を三つだけ選び、簡単な手順を書き出す。ここまでできれば、第2章の目的は果たせている。

今週やることは、依頼の状態欄と優先順位ルールを実際の案件へ当てはめ、記録が回るか試すことである。口頭依頼やチャット直投げが残るなら、どこで崩れるかも確認したい。

後でやることは、退職対応、権限変更、端末配布のような定型業務を、申請書やチェックリストへ少しずつ落とすことである。最初から制度化しなくてもよいが、窓口と台帳だけは流れの中心に置いた方がよい。

第3章

# ひとり情シスの仕事を回す基本原則

朝の時点では、今日は比較的落ち着いていると思っていた。ところが、出社して 30 分で、プリンタ設定の相談、退職者アカウント停止の依頼、来週導入する SaaS の権限設計確認、Wi-Fi の不調、ライセンス追加の問い合わせが一気に来る。どれも急ぎに見える。とりあえず目の前から片づけようとして、チャットに返事をし、管理画面を開き、別の依頼を思い出して中断し、メールを返し、また最初の作業に戻る。気づけば夕方になり、どれも終わっていない。

ひとり情シスの仕事は、件数が多いだけで苦しくなるのではない。仕事の回し方に原則がないまま、すべてを同じ重さで受けてしまうと苦しくなる。標準化できる仕事まで毎回個別判断し、緊急と重要を区別せず、手を付ける案件を増やしすぎ、自分しか知らない状態を放置すると、どれだけ頑張っても仕事は軽くならない。

第2章では、窓口、記録、台帳、手順、優先順位という土台を整えた。この章では、その土台の上で仕事をどう回すかを扱う。ここで必要なのは、気合いではなく原則である。

## まず原則を持つ

ひとり情シスの現場では、毎日違う問題が起きているように見える。しかし、仕事の回し方として必要な原則はそれほど多くない。本書では、まず次の五つを基本原則とする。

第一に、標準化できるものは早く標準化する。

第二に、例外だけを個別に考える。

第三に、優先順位は緊急度と重要度で決める。

第四に、同時に抱える仕事を増やしすぎない。

第五に、自分で持つ仕事と外に出す仕事を分ける。

この五つがあると、すべての仕事を自力で瞬間判断し続ける状態から抜けやすくなる。逆に、この五つがないと、仕事はいつまでも「来た順」「近い順」「強く言われた順」に流される。

ここで大事なのは、原則は自分を縛るためのものではなく、自分を守るためのものだということである。ひとり情シスは、頼まれれば何でも対応してしまいやすい。だからこそ、頼まれ方ではなく、回し方で仕事を決める必要がある。

## 標準化できるものを早く標準化する

ひとり情シスの仕事には、毎回判断が必要なものと、実は毎回同じように処理できるものが混ざっている。たとえば、パスワード再設定、ライセンス追加、アカウント発行、端末キッティング、共有フォルダ作成、退職時の基本停止作業などは、ある程度まで定型化できる。

Atlassian の service request management が示す通り、こうした recurring request は repeatable procedure で処理する方がよい。つまり、毎回考えない方がよい。毎回考えている限り、同じ仕事に同じ時間を払い続けることになる。

ここで必要なのは、完璧な標準化ではない。まずは、「これは毎回ほぼ同じだ」と気づくことだ。そのうえで、受付条件、必要情報、処理手順、完了条件を決める。例外が出たら、その例外だけを考えればよい。

たとえば、新しいソフトを入れてほしいという依頼を考えてみる。毎回その場で「入れてよいか」「誰が承認するか」「ライセンスはあるか」「端末要件は合うか」を考えていると、依頼ごとに判断疲れが起きる。そこで、申請時に必要な情報を決め、承認者を決め、標準的な導入条件を決めておけば、例外だけに頭を使える。

変更作業も同じである。すべての変更を重い仕事として扱う必要はない。日常的で影響範囲が読みやすい変更は、標準変更として扱いやすい。逆に、全社に影響する設定変更や、切り戻しが難しい変更は、慎重に扱うべきである。毎回全部を大仕事にすると、変更が滞る。全部を軽く扱うと事故になる。だから、重み付けが必要になる。

標準化の目的は、仕事を機械的にすることではない。判断を減らすことで、本当に考えるべき仕事へ時間を回すことである。

## 緊急度と重要度を分ける

ひとり情シスの仕事で最も崩れやすいのは、緊急と重要が混ざることである。今すぐ返事が必要に見える依頼が、会社にとって本当に重要とは限らない。逆に、今すぐ困っている人は少なくても、後回しにすると大きな事故になる仕事もある。

ここで分けて考えたいのが、緊急度と重要度である。

緊急度は、どれだけ早く処理しなければならないかを示す。たとえば、全社 VPN 障害、退職当日のアカウント停止、役員会直前の会議室接続不良などは緊急度が高い。

重要度は、事業やリスクへの影響の大きさを示す。たとえば、管理者権限の棚卸し、契約更新日の整理、バックアップ復元確認、共有アカウントの見直しは、今この瞬間に困る人が少なくても重要度が高い。

問題は、緊急度が高い仕事ばかりを追い続けると、重要だが緊急ではない整備が永遠に進まないことである。そして、その整備が進まないからこそ、緊急案件が増え続ける。

優先順位を付けるときは、少なくとも次の四つで見るとよい。

- 何人に影響するか
- 重要業務に影響するか
- セキュリティや法務のリスクがあるか
- 締切や即日性があるか

この軸で見れば、声の大きい依頼より先にやるべき仕事が見えやすくなる。優先順位は、自分の気分で決めるものではない。仕事の性質で決めるものである。

## 同時に抱える仕事を減らす

仕事が終わらない原因の一つは、件数そのものより、途中で止まっている仕事が増えすぎることである。Atlassian の Kanban が示す WIP limit の考え方は、ひとり情シスにもよく当てはまる。手を付けた仕事を増やすほど、切り替えコストが増え、どれも終わりにくくなる。

たとえば、五つの仕事を少しずつ進めるより、まず二つを終わらせた方が、全体の流れは良くなることが多い。にもかかわらず、ひとり情シスは依頼が来るたびに反応しやすい。その結果、「対応中」が増え続ける。

これを防ぐには、進行中案件の上限を意識的に決めるのがよい。厳密なルールでなくてよい。「今は三件までを進行中にし、それ以上は待ち行列へ置く」と決めるだけでも、かなり違う。緊急障害のように割り込みが必要なものは別として、通常の依頼や改善作業は、抱えすぎない方が早く終わる。

ここで重要なのは、「忙しいから全部同時にやる」は、実際には解決策になっていないということだ。忙しい時ほど、途中の仕事を減らし、終わらせる順序を決める必要がある。

## 火消しだけで終わらせない

ひとり情シスの仕事は、放っておくと火消し中心になる。Wi-Fi　が遅い。印刷できない。ライセンスが足りない。同じ人から何度も同じ問い合わせが来る。毎回その場で直して終わりにしていると、確かに目の前の問題は消える。しかし、仕事は減らない。

Atlassian の problem management が扱うのは、こうした再発問題の根本原因である。ひとり情シスが大掛かりな problem management を導入する必要はないが、最低限、「繰り返し起きる問題」を別で残すだけでも意味がある。

たとえば、毎月同じ VPN 接続問い合わせが来るなら、手順書や FAQ に落とせないかを見る。同じ部署で権限申請の不備が繰り返されるなら、申請フォームや案内文を見直す。特定の会議室だけ接続トラブルが続くなら、機器構成や利用方法を点検する。こうして一段深く見ると、応急処置で終わっていた仕事が、再発防止の対象へ変わる。

火消しだけで終わらせないとは、毎回原因分析会をするという意味ではない。少なくとも、「また起きた」を記録し、同じ仕事を減らす方向へ一歩進めるという意味である。

## 何を自分で持ち、何を外に出すか

ひとり情シスが苦しくなるもう一つの理由は、「自分がやるべき仕事」と「自分が持つ必要はない仕事」が混ざっていることである。ここで外注を考えると、「自分が苦手な仕事を投げること」と誤解されがちだが、実際にはそれだけではない。

仕事を外に出すかどうかは、少なくとも次の四つで考えるとよい。

一つ目は頻度である。年に一回しか触らないのに、失敗すると影響が大きい作業は、外部支援を使った方が安全なことがある。

二つ目は専門性である。ネットワーク機器の高度な障害解析や、特殊な法務対応のように、深い専門知識が必要なものは、常時内製する価値が低い場合がある。

三つ目は事故影響である。ミスした時の被害が大きい作業は、自社での理解が必要か、外部の専門家に頼るべきかを慎重に考える。

四つ目は自社理解の必要性である。入社、異動、退職、権限設計、現場の業務フローの理解が必要なものは、自社側で握っていた方がよいことが多い。

ここで大切なのは、外注は責任放棄ではないということだ。NIST の C-SCRM が示すように、要求と責任分界を持たずに頼めば、単なる丸投げになる。外部へ出すなら、「何を依頼するか」「何を自社で確認するか」「障害時に誰が判断するか」を明確にしなければならない。

つまり、外に出すべき仕事とは、自社で一から抱える価値が低く、外部の専門性や継続保守の方が効果的な仕事である。一方で、会社の業務理解や日々の判断が必要な仕事は、自社で持つ価値が高い。

## 自分が最大のボトルネックにならない

ひとり情シスは、放っておくと自分自身が最大のボトルネックになる。自分しか知らない管理者設定、自分しか見られない契約情報、自分しか触れない共有メール、自分しか分からない手順。これらは、本人が優秀であるほど増えやすい。

しかし、これは強さではなく、構造的な弱さである。自分が休んだ時、退職した時、緊急対応中で手が離せない時に、一気に詰まるからだ。

Kanban が示す overlapping skill sets の考え方を、ひとり情シスの現場向けに言い換えるなら、「自分しかできない仕事を減らす」ことである。すべてを誰でもできるようにする必要はない。だが、少なくとも次のものは残したい。

- 緊急連絡先
- 管理者権限の所在
- よく使う手順
- ベンダー連絡先
- 契約更新の確認方法

ひとり情シスの仕事を回すとは、自分が頑張って回すことではない。自分が詰まっても、全部は止まらない状態を少しずつ作ることでもある。

## 通常時の例と、崩れた時の例

ここで、同じ会社でも回し方でどう差が出るかを見てみよう。

通常時の例では、新規ソフト導入の依頼が来た時、申請フォームで用途、利用者、端末、予算、承認者を確認する。標準条件に合うものはそのまま処理し、例外だけを確認する。進行中案件は三件までに抑え、緊急障害がなければ順番に処

理する。よく来る問い合わせは手順化し、毎月繰り返す問題は　FAQ　に落とす。ネットワーク機器の高度障害だけは外部保守へ即連絡する。結果として、目の前の依頼を全部すぐ処理できなくても、仕事は前へ進む。

崩れた時の例では、依頼が来た順に全部へ反応する。ソフト導入の基準がなく、その場で判断する。進行中案件は常に十件以上あり、どれも途中で止まる。問い合わせが再発しても記録がなく、毎回一から説明する。ネットワーク障害が起きた時の外部連絡先が分からず、自分が調べ始める。結果として、本人は一日中忙しいのに、重要な仕事ほど進まない。

この差を生むのは、能力差より原則の差である。

## 最低限ここまではやる

第3章の段階では、次の六つができれば十分に前進である。

一つ目。定型作業と例外作業を分けて考える。

二つ目。緊急度と重要度を区別して仕事を並べる。

三つ目。進行中案件の数を意識的に制限する。

四つ目。繰り返し起きる問題を別で残す。

五つ目。外に出せる仕事を一つでも見つける。

六つ目。自分しか知らない情報を一つでも減らす。

これだけでも、仕事は「何でも即応する状態」から「流れを設計する状態」へ変わり始める。

**確認したいこと**

- 定型作業と例外作業を分けている

- 優先順位を緊急度と重要度で説明できる
- 進行中案件の件数を把握している
- 繰り返し起きる問題を記録している
- 外注候補を一つ以上挙げられる
- 自分しか知らない設定や情報を減らしている

## 今日、今週、後でやること

今日やることは三つでよい。まず、今抱えている仕事を一覧にして、定型と例外に分ける。次に、進行中案件の上限を一つ決める。最後に、自分しか知らない情報を一つだけ文書化する。

今週やることは、標準化できる作業を三つ選び、手順化することだ。加えて、繰り返し起きている問い合わせや障害を一つ選び、再発防止の観点で見直す。

後で整えることは、外注先の見直しや、自動化候補の整理である。ここは今すぐ全部やらなくてよいが、自分で持ち続けるべき仕事かどうかは意識しておきたい。

第II部

# 経営、統制、調達、導入

判断の所在と責任分担を曖昧にしない。

第4章〜第7章

第4章

# 経営と IT ガバナンス

退職者のアカウント停止が遅れる。バックアップの復元確認は、気になっているが後回しになっている。来月更新の SaaS は、費用を続けるべきか判断が付かない。ネットワーク機器の保守を延長するかも決まっていない。ひとり情シスは、それぞれに問題があることを知っている。しかし、どこまで自分が決めてよいのか分からない。経営へ相談しても、「それって IT の話でしょう」と返される。結局、自分で抱えたまま時間だけが過ぎる。

この状態は、現場の努力不足ではない。会社として、何を守り、何を優先し、誰が何を決めるかが曖昧だから起きる。ひとり情シスがどれだけ真面目でも、経営判断が必要なことまで一人で抱え込めば、いつか必ず限界が来る。

この章で扱う IT ガバナンスとは、重い制度や大きな委員会のことではない。小さな会社でも必要な、「判断の所在を明確にし、現場の運用を経営とつなぐ仕組み」のことである。経営と IT ガバナンスを整えると、ひとり情シスの仕事は急に減らないかもしれない。しかし、少なくとも「自分だけが不安を知っている」状態からは抜けやすくなる。

## IT ガバナンスとは何か

IT ガバナンスという言葉は、実態より大げさに聞こえやすい。だが、本質は単純である。会社として、IT と情報セキュリティに関して、誰が何を決めるかを明確にすることだ。

NIST Cybersecurity Framework 2.0 が 2024年2月26日に公開された時、注目されたことの一つが `Govern` を独立機能として置いた点である。これは、保護策だけでは十分ではなく、組織の文脈、リスク管理方針、役割、監督を先に整える必要があるという考え方を示している。現場で何をするかの前に、会社として何を重視し、どこまでのリスクを受け、誰が責任を持つのかが必要だということだ。

だから、ガバナンスは会議を増やすことではない。承認欄を増やすことでもない。現場に丸投げしないこと、そして現場が一人で抱え込まなくてよい状態を作ることだ。

ひとり情シスの現場では、ガバナンスがないと、現場担当が次のような判断まで背負いやすい。

- この契約更新は続けるべきか
- この障害リスクは受け入れるのか
- バックアップ強化に費用を使うのか
- 外部委託へ切り替えるのか
- どの業務を優先的に守るのか

これらは、技術判断だけではない。本来は会社の判断である。ガバナンスが必要なのは、そこを分けるためだ。

## 経営が持つべき論点と、現場が持つべき論点

ひとり情シスが楽になるためには、「何でも経営に聞く」状態を作ればよいわけではない。逆に、「現場で全部決める」状態も危うい。必要なのは、経営が決めるべきことと、現場が決めるべきことを分けることである。

経営が持つべき論点は、大きく四つある。

第一に、何を優先して守るかである。売上、受発注、顧客対応、給与、法令対応など、止まって困る業務の優先順位は経営判断に属する。

第二に、どのリスクをどこまで受けるかである。古い機器をすぐ更新できない、すべての運用を即時に強化できない、という現実はある。その時に、何を許容し、何を許容しないかは会社として決める必要がある。

第三に、どこへ投資するかである。MFA 導入、バックアップ強化、端末更新、外部保守、監査ログ強化などは、費用と優先順位を伴う。

第四に、どこまでを外部へ任せるかである。ネットワーク保守、監視、ヘルプデスク、法対応支援など、外部委託の方針は現場だけでは決めきれない。

一方で、現場が持つべき論点もある。設定手順、日常運用、問い合わせ対応、標準手順、軽微な改善、日々の記録は現場で回すべきだ。毎回経営判断を仰いでいては運用が止まる。

ここで重要なのは、ひとり情シスが「何を自分で決めてよいか」を把握することだ。経営判断が必要な論点まで自分で抱えると、判断の重みが増すだけでなく、後で「なぜその判断をしたのか」を説明しにくくなる。

## 何を守るのかを整理する

ガバナンスの出発点は、制度でも方針書でもなく、「何を守るのか」を整理することにある。守る対象が曖昧なままでは、優先順位も報告も作れない。

ここで最初に見たいのは、重要業務である。たとえば、受発注、売上計上、請求、顧客サポート、給与計算、在庫管理、店舗運営など、止まると事業に直接影響する業務を挙げる。全部を同じ重さで扱う必要はない。まずは三つから五つに絞る。

次に、その業務を支える重要資産を見る。どのシステム、どのデータ、どの端末、どのアカウント、どの外部サービスが止まると業務が止まるかを確認する。ここで初めて、「この SaaS は便利だから重要」ではなく、「この業務を止めないために重要」という見方ができる。

さらに、その業務と資産に対して、どんなリスクがあるかを考える。アカウント停止漏れ、契約更新漏れ、バックアップ不足、外部共有ミス、ネットワーク障害、ベンダー依存、管理者権限の属人化。こうして見ていくと、技術課題が事業課題として並び始める。

たとえば、受発注業務が重要なら、それを支えるメール、クラウドストレージ、認証、ネットワークの安定性が重要になる。給与計算が重要なら、人事データ、委託先、アクセス権、バックアップが重要になる。このつながりが見えると、経営へも説明しやすくなる。

## 方針と優先順位を決める

何を守るかが見えたら、次は方針と優先順位である。ここで難しく考える必要はない。NIST の Organizational Profiles や IPA の可視化ツール、DX 推進指標に共通しているのは、現状と目標の差を見えるようにすることだ。

まず、現状を書く。たとえば「管理者権限の一覧がない」「退職者対応のチェックリストがない」「バックアップはあるが復元確認をしていない」「契約更新管理が担当者依存している」といった状態である。

次に、目標を書く。たとえば「三か月後には管理者権限一覧がある」「退職対応チェックリストを運用している」「重要システムの復元確認を年一回実施している」「契約更新日を一覧で見られる」といった状態である。

最後に、その差の中から優先順位を決める。ここで全部を一気に埋めようとしないことが大切だ。重要業務に直結するもの、事故時の影響が大きいもの、すぐ着手できるものから順に並べる。

この時に役立つのが、短い基本方針である。大きな規程集をいきなり作る必要はない。SECURITY ACTION 二つ星でも、まず自己診断と情報セキュリティ基本方針を出発点にしている。小さな会社なら、まずは次のような短い方針で十分だ。

- 重要業務を止めないことを優先する
- アカウントと権限は記録して管理する
- 退職、契約更新、バックアップは期限管理する
- 事故や重大リスクは経営へ報告する

方針が短くても、会社として合意されていることに意味がある。ひとり情シスの個人ルールを、会社のルールへ変える最初の一歩になるからだ。

## 経営へどう報告するか

ガバナンスが弱い会社では、報告も崩れやすい。技術用語ばかり並び、経営が何を見ればよいか分からないか、逆に「特に問題ありません」だけで終わってしまうかのどちらかになりやすい。

NIST の ERM Quick-Start Guide が 2024年10月21日に示したように、重要なのはサイバーリスク情報を enterprise risk management に統合することだ。言い換えれば、技術の出来事を経営が判断できる言葉へ翻訳する必要がある。

経営への報告は、一枚でよい。たとえば、次の四項目だけでも十分に意味がある。

- 今月の出来事
  - 重大障害、事故、更新、棚卸し結果
- 今ある主なリスク
  - 重要業務に影響する未対応事項
- 近い期限
  - 契約更新、証明書更新、保守期限、重要な見直し時期
- 判断が必要なこと
  - 投資、許容リスク、委託、優先順位変更

ここで大切なのは、報告の主語を技術から事業へ移すことだ。たとえば「VPN機器が古い」ではなく、「在宅勤務と拠点接続の停止リスクが高いので、更新判断が必要」と言う。「管理者アカウントの棚卸しが未了」ではなく、「退職や権限漏れの事故リスクが残っている」と言う。この翻訳ができると、経営は初めて判断しやすくなる。

## 小規模組織で現実的に回るガバナンス

ここまで読むと、ガバナンスは手間がかかりそうに見えるかもしれない。だが、小規模組織で必要なのは大掛かりな委員会ではない。むしろ、重い制度は続かない。

現実的なのは、月一回三十分でもよいので、定例の場を持つことだ。参加者は、経営者か責任者、ひとり情シス、必要に応じて総務や経理で十分である。そこで毎回同じ型で確認する。

- 重要業務に関わる問題は何か
- 期限が近いものは何か
- 今月起きた出来事は何か
- 判断が必要なことは何か
- 次に優先する整備は何か

この型があるだけでも、ひとり情シスが一人で不安を抱える状態はかなり減る。さらに、IPA の可視化ツールや DX 推進指標の考え方を借りて、現状と目標を年に一度でも見直せば、会社としての進み方が見える。

重要なのは、会議の長さではなく、見るべきものと決めるべきものが定まっていることだ。逆に、定例の場があっても、何を見て何を決めるかが曖昧なら、ガバナンスにはならない。

## 通常時の例と、崩れた時の例

通常時の例では、月初にひとり情シスが一枚報告を作る。内容は、先月の障害、今月の更新期限、残っている主なリスク、判断が必要な事項の四つだけである。月一回、社長と管理部門責任者と三十分話す。そこで、「今月は退職者対応フロー整備を優先する」「VPN 機器更新は来期予算へ入れる」「契約更新管理は総務と分担する」といった判断を決める。すべてを解決できなくても、少なくとも会社としての優先順位がそろう。

崩れた時の例では、ひとり情シスが不安だけを抱えている。更新期限が近い契約があるが、誰に相談すべきか分からない。退職者対応の漏れが気になるが、業務フローを変える権限がない。バックアップの弱さも分かっているが、費用を使う判断は取れない。結果として、現場の不安は増え、経営は何も知らず、事故が起きた時だけ「なぜ言わなかったのか」となる。

この差を生むのは、立派な制度の有無ではない。判断の所在と報告の型があるかどうかである。

## よくある失敗

第4章で特に避けたい失敗は五つある。

一つ目は、経営に何も上げないことだ。現場で抱えた方が早い場面はあるが、経営判断が必要な論点まで抱え込むと、後で必ず詰まる。

二つ目は、逆に何でも経営へ上げることだ。日常運用まで上げていては、現場が止まる。役割分担が必要である。

三つ目は、報告が技術用語だらけになることだ。経営は技術詳細を知らなくてよい。必要なのは、事業影響と判断事項である。

四つ目は、重要業務を整理しないことだ。これがないと、優先順位が永遠にぶれる。

五つ目は、会議だけ増やして何も決めないことだ。ガバナンスは資料作りのためにあるのではなく、意思決定のためにある。

## 最低限ここまではやる

第4章の段階では、次の五つができれば十分に前進である。

一つ目。止められない業務を三つ書き出せること。

二つ目。経営が決めるべき論点と、現場で回す論点を分けられること。

三つ目。短い基本方針を持てること。

四つ目。月次で共有する一枚報告の型を持てること。

五つ目。月一回でも、経営と IT を話す場があること。

これだけでも、現場の孤立はかなり減る。ひとり情シスの仕事を安定させるとは、現場が全部強くなることではない。会社として判断に参加することでもある。

**確認したいこと**

- 重要業務を三つ以上挙げられる
- その業務を支える重要システムや資産を説明できる
- 経営判断が必要な論点を言語化できる
- IT 課題を事業影響で説明できる
- 月次で見る項目を決めている
- 現状と目標の差を説明できる

## 今日、今週、後でやること

今日やることは三つでよい。まず、止められない業務を三つ書く。次に、今気になっている IT やセキュリティ上の不安を、事業影響の言葉へ書き換える。最後に、月次一枚報告のひな型を作る。

今週やることは、経営者か責任者と話す時間を確保することだ。長い会議は要らない。三十分でよいので、重要業務、残っているリスク、必要な判断を共有する。

後で整えることは、方針の文書化、可視化ツールの活用、目標設定の見直しである。最初から全部を制度化する必要はないが、現場の不安を会社の判断へ変える流れだけは早めに作りたい。

第5章

# IT 予算、契約、調達、ライセンス

一番安いプランを選んだ。営業からは、今なら割引が大きいと言われた。機能も一通りそろっているように見えた。だから導入した。ところが、数か月後から空気が変わる。設定変更のたびに追加費用が発生する。問い合わせても、そこはサポート対象外だと言われる。人員が減っても年間契約なので席数を減らせない。退職者のアカウントを止めたのに課金は続いている。解約しようとすると、データの取り出しや移行に思った以上の手間がかかる。

こうした詰まり方は珍しくない。むしろ、ひとり情シスの現場ではよく起きる。理由は単純である。調達を「買う瞬間」の話として見てしまい、「使い続ける」「増減する」「引き継ぐ」「やめる」まで含めて考えないからだ。

第4章では、経営と現場の間で何を守り、何を優先し、誰が何を決めるかを整理した。第5章は、その判断を具体的な購入と契約に落とす章である。ここで扱うのは節約術ではない。後で運用が破綻しないための判断軸である。IT の予算、契約、調達、ライセンスを別々の話としてではなく、一つの流れとして見る。

## 調達は買い物ではなく設計である

ひとり情シスの現場では、調達が「どの製品を選ぶか」という話に縮みやすい。だが、本当はもっと広い。調達とは、導入後の運用のしやすさ、障害時の支援、権限管理、費用変動、契約更新、解約や移行まで含めて、自社がその仕組みを回せるかを設計することである。

IPA　の情報システム・モデル取引や契約の考え方でも、重要なのは責任分界の明確化である。つまり、契約は「発注したら終わり」の紙ではなく、誰が何をするかをはっきりさせるための道具だということだ。ここが曖昧なまま導入すると、障害時に「そこは御社側の作業です」と言われ、変更時には「別料金です」となり、更新時には「その条件では減数できません」となる。

価格が安いかどうかは、その一部でしかない。むしろ、価格だけで決めた時ほど、後で高くつく。人の手間、問い合わせ回数、設定の複雑さ、移行の難しさ、解約のしにくさまで含めて初めて調達の判断になる。

第5章でまず持っておきたい視点は一つだけでよい。何を買うかではなく、買った後に自社で回るかを見る。これが調達の出発点である。

## IT 予算は価格ではなく継続コストで考える

予算の話になると、どうしても初期費用や月額費用の比較に意識が寄りやすい。しかし、ひとり情シスが本当に見るべきなのは、払う金額そのものより、その支出が何を減らし、何を増やすかである。

たとえば、月額が安い SaaS でも、設定変更のたびにベンダーへ依頼が必要なら、社内の待ち時間と調整負荷が積み上がる。サポートが弱ければ、結局自分で調べて回る時間が増える。逆に、少し高くても権限管理や監査ログや自動化が整っていれば、日常運用の手間は下がる。予算とは単価の話ではなく、手間と事故と停止リスクまで含めた継続コストの話である。

ここで見たいのは、少なくとも次の四つである。

- 初期費用
- 月額や年額の継続費用
- 追加費用が発生する条件

- 自社の運用工数

このうち、最後の運用工数は見落とされやすい。だが、ひとり情シスにとっては非常に大きい。担当者が一人しかいない環境では、「毎月少し面倒」が積み上がるだけで、十分に重いコストになるからだ。

だから予算を考える時は、「いくらかかるか」だけでなく、「この製品を選ぶことで毎月何時間減るか、逆に何時間増えるか」を考える必要がある。金額に換算しなくてもよい。だが、手間が増えるか減るかは必ず見たい。

もう一つ重要なのは、重要業務との関係である。第4章で整理した重要業務に直結するシステムなら、安さだけで選ぶべきではない。受発注、売上計上、顧客対応、給与計算のような止まると困る業務では、安いことより、止まりにくいこと、復旧しやすいこと、相談しやすいことの方が重い場合が多い。

## 見積比較では何を見るか

見積を取ると、金額の列が一番目に入りやすい。だが、比較表を作るなら、金額列だけでは足りない。最低でも、次の軸を並べたい。

- 機能
- 非機能
- サポート範囲
- 契約期間と更新条件
- 解約や減数のしやすさ
- データ移行と持ち出しのしやすさ
- 追加課金の条件

機能は分かりやすい。欲しいことができるかどうかだ。しかし、実際に差が出やすいのはその先である。たとえば、同じように見えるクラウドサービスでも、障害時の連絡手段、問い合わせ窓口、回答速度、バックアップの扱い、監査ログの有無、権限の細かさ、API の使いやすさはかなり違う。

また、契約期間の違いは非常に大きい。Google Workspace の公式案内でも、柔軟に席数を増減できる契約と、年間コミットが必要な契約では、運用の自由度が大きく異なる。人員変動が大きい会社で単価の安い年間契約を選ぶと、結果的に使わない席の費用を払い続けることがある。単価の安さだけを見ると得に見えても、増減のしにくさまで入れると逆転する。

見積比較で大切なのは、「導入時の一回勝負」で表を作らないことだ。日常運用、組織変更、障害、契約更新、解約の各場面で何が起きるかを想像して並べる。比較表は、製品選定のためだけではなく、後で自分を守るための記録でもある。

## 非機能要件を先に確認する

調達で詰まりやすいのは、機能不足より非機能の見落としである。IPA の非機能要求グレードや要求策定ガイドが示している通り、業務で使うシステムを選ぶ時は、できることだけでなく、止まりにくさ、支えやすさ、管理しやすさを見なければならない。

非機能という言葉は抽象的に見えるが、ひとり情シスの実務に引き直すと分かりやすい。見るべきことは、たとえば次のような項目である。

- 障害が起きた時にどこまで支援されるか
- バックアップや復旧はどうなっているか
- ログはどこまで残るか
- 権限を細かく分けられるか

- 管理画面は一人で扱える複雑さか
- 外部連携やデータ出力はしやすいか
- 保守や変更作業に追加費用が発生するか

たとえば機能面では十分でも、監査ログが弱く、管理者権限の分離もできず、退職者データの保持も難しい製品なら、後で確実に困る。逆に、機能が少し地味でも、権限、ログ、データ出力、サポートがしっかりしていれば、長く安定して使いやすい。

ひとり情シスが非機能を見る理由は、立派な要件定義書を書くためではない。後で自分だけが無理をしないためである。導入後の運用を一人で背負う可能性が高い以上、運用負荷を先に見ておくのは当然の防御である。

## 契約で確認すべきポイント

契約は法務部門だけの話ではない。もちろん法的な詳細条項を全部一人で判断する必要はない。だが、現場として確認すべき論点はある。それは、この契約で何を期待でき、何を期待できないかを明確にすることだ。

最低でも確認したいのは、次のような点である。

- 誰が何をする契約なのか
- サポート対象はどこまでか
- 問い合わせ方法と受付時間はどうなっているか
- 障害時の連絡やエスカレーションはどうなるか
- データの保存場所や取り出し条件はどうなっているか
- 自動更新の有無と解約期限はどうなっているか
- 再委託や外部サービス利用の扱いはどうなっているか

ここで重要なのは、期待と現実のずれを潰すことである。たとえば「導入支援あり」と書かれていても、実際は初期設定の説明だけで、データ移行や運用設計は対象外かもしれない。「サポートあり」と書かれていても、メールのみで回答に数日かかるかもしれない。「バックアップあり」と書かれていても、利用者側が復元依頼できる範囲は限定的かもしれない。

また、解約条件は後回しにされやすいが、むしろ導入前に見ておきたい。どのくらい前までに連絡が必要か。契約期間中の減数はできるのか。データはどの形式で返せるのか。終了後の保持期間はどうなっているのか。これらは、やめる時だけでなく、価格交渉や継続判断にも効いてくる。

## ライセンス管理は調達の一部である

ライセンス管理は、購入後の事務作業のように見えやすい。だが実際には、調達の時点で設計しておかないと後で苦しくなる。なぜなら、ライセンス管理は課金管理だけでなく、権限管理、人の出入り、データ保持とつながっているからだ。

Microsoft 365 でも Google Workspace でも、ライセンスは誰にどう配るかで運用負荷が変わる。個別に手で割り当てるのか。グループベースで付与するのか。新規入社時に自動で付くのか。不要になった時にどの手順で外すのか。こうした設計を曖昧にしたまま使い始めると、増員時に毎回作業がばらつき、棚卸し時には未使用ライセンスが見つからず、退職時には権限と課金の両方で漏れが出る。

ひとり情シスの現場では、ライセンス管理を次の三つで同時に見たい。

- 課金
- 権限
- データ保持

課金だけを見て削減すると、必要なデータまで失うかもしれない。権限だけを見て管理者を増やすと、ライセンス操作の統制が崩れるかもしれない。データ保持だけを優先すると、不要な高額ライセンスを抱え続けるかもしれない。だから、この三つを分けずに考える。

また、誰がライセンスを操作できるかも重要である。Microsoft の公式資料でも、必要最小限の権限を使う考え方が明示されている。ひとり情シスが全部を持つ場面は多いが、それでも高権限をむやみに広げない原則は持っておきたい。

## 停止、削除、アーカイブを混同しない

ライセンス運用で特に危ないのが、「止めたから費用も止まるだろう」「削除したからきれいに終わっただろう」と考えてしまうことだ。実際には、停止、削除、アーカイブは意味が違う。

Google Workspace の公式説明では、停止した利用者でも課金が続く場合がある。つまり、ログインできない状態にしたことと、費用が減ることは同じではない。また、ライセンスを外したり利用者を削除したりすると、データ保持や復元可能期間に影響する。保持したいデータがあるなら、削除よりアーカイブの方が適切なこともある。

ここで整理したいのは、三つの問いである。

- その人を今すぐ使えなくしたいのか
- その人のデータを残したいのか
- その費用を止めたいのか

この三つは似ているようで別の話である。入社、異動、休職、退職のたびにここを混同すると、課金漏れ、データ消失、権限残りのどれかが起きやすい。第9章では人の出入りの運用を詳しく扱うが、第5章ではその前提として、「費用」「利用可否」「保持」は同じ操作ではないと理解しておきたい。

## 小さな会社で現実的に回る判断

ここまでを読むと、調達前に確認することが多すぎるように見えるかもしれない。だが、小さな会社で必要なのは完璧な調達プロセスではない。後で詰まりやすい点を、最低限先に見ておくことだ。

現実的には、まず次の四つだけでも十分に効果がある。

- 固定費になっている契約を一覧にする
- 更新日と解約期限をカレンダーへ入れる
- 見積比較表に非機能と解約条件の列を足す
- 未使用ライセンスと不要契約を定期的に見る

この四つがないと、判断はいつも場当たりになる。逆に、この四つがあるだけで、「どれを続けるか」「どれを減らすか」「どれを標準にするか」をかなり落ち着いて考えられる。

もう一つ大事なのは、標準製品を決めることである。部門ごとに似たようなツールが乱立すると、ライセンス管理、問い合わせ対応、教育、権限管理が全部重くなる。全部を一つに統一する必要はないが、メール、ストレージ、チャット、会議、端末管理のような基盤部分は、できるだけ標準を持った方がよい。標準化は第3章で扱った運営原則の延長であり、調達でも同じように効く。

## 通常時の例と、崩れた時の例

通常時の例では、新しい SaaS を検討する時に、ひとり情シスが三社比較表を作る。列は金額だけではなく、サポート、ログ、権限、データ出力、契約期間、減数条件、解約期限、管理画面の扱いやすさまで入っている。経営へは「A が最安だが、減数しにくく運用負荷が高い。B は少し高いが、権限管理と解約条件が良い。重要業務に使うので B を推奨する」と説明する。導入後は、ライセンス付与方法、棚卸しの頻度、退職時の扱いまで最初に決める。結果として、日常運用が安定する。

崩れた時の例では、営業提案の勢いで一番安い年間契約を選ぶ。比較表は金額と機能だけで、減数条件もデータ移行条件も見ていない。入社時は都度手作業でライセンスを付け、退職時は停止だけして終える。半年後、使っていないライセンスが積み上がり、解約したいが更新期限を過ぎており、古いデータをどう残すかも決まっていない。ひとり情シスは、安く入れたはずの製品の後始末に時間を取られ続ける。

この差は、特別な調達部門の有無ではない。買う前に、後で起きる運用を想像したかどうかである。

## よくある失敗

第5章で特に避けたい失敗は五つある。

一つ目は、単価だけで決めることだ。安いことは大事だが、安さだけでは十分ではない。

二つ目は、機能だけを見て非機能を見ないことだ。ログ、権限、バックアップ、サポート、移行性を後から足すのは難しい。

三つ目は、契約を読まずに運用を始めることだ。特に自動更新、減数条件、解約期限、支援範囲は必ず見たい。

四つ目は、ライセンス管理を総務作業のように軽く見ることだ。実際には課金、権限、データ保持に直結する。

五つ目は、停止、削除、アーカイブを同じものとして扱うことだ。ここを混同すると、費用かデータか統制のどこかで事故になる。

## 最低限ここまではやる

第5章の段階では、次の五つができれば十分に前進である。

一つ目。契約中の主要 SaaS と保守契約を一覧にできること。

二つ目。各契約について、更新日と解約期限を把握していること。

三つ目。新しい製品を比較する時に、金額以外の列を持てること。

四つ目。未使用ライセンスと退職者アカウントの状態を確認できること。

五つ目。停止、削除、アーカイブの違いを説明できること。

これだけでも、調達はかなり安定する。ひとり情シスの仕事を守るとは、導入時に頑張ることではない。導入後に無理をしなくて済む形を先に作ることでもある。

**確認したいこと**

- 単価以外の比較軸を持っている
- 非機能要件を確認している
- 契約更新日と解約期限を把握している
- サポート範囲と責任分界を説明できる

- ライセンス棚卸しの方法を持っている
- 停止、削除、アーカイブの違いを説明できる

## 今日、今週、後でやること

今日やることは三つでよい。まず、現在契約している主要 SaaS と保守契約を一覧にする。次に、その中で更新日が近いものと、使っていない可能性があるライセンスを一つずつ洗う。最後に、今後の見積比較表へ「非機能」「解約条件」「データ移行」の列を足す。

今週やることは、主要契約について、更新条件、減数条件、サポート範囲、解約期限を確認することだ。可能なら、経営や責任者とも一度共有し、「単価ではなく運用まで見て選ぶ」という基準をそろえておきたい。

後で整えることは、ライセンス付与の標準化、契約更新カレンダーの運用、標準製品の整理である。これらは一度に完成しなくてよいが、放っておくほど後で整理が難しくなる。

第6章

# ベンダー管理と外部委託

ネットワーク保守は外部へ任せている。SaaS　の初期設定は導入会社が入った。PC のキッティングも一部は業者に頼んでいる。だから自分の仕事は軽くなるはずだった。ところが実際には、障害が起きると、どこへ連絡すべきかを自分が判断する。設定変更の相談が来ると、どこまで業者へ頼めるのかを自分が確認する。契約更新が近づけば、継続するかどうかを自分が考える。業者が複数いれば、「そこは別ベンダーの範囲です」と言われ、そのつなぎ役も自分になる。

ひとり情シスにとって、外部委託は仕事をなくす魔法ではない。むしろ、うまく使えば自分だけでは回し切れない部分を補えるが、使い方を誤ると、見えない仕事と調整仕事を増やす。

第5章では、何を買い、どう契約し、どうライセンスを持つかを扱った。第6章で扱うのは、その次である。契約した相手と、どう付き合い、どう監督し、どう終わらせるかだ。ここで言うベンダー管理は、相手を疑うための仕組みではない。自社に残る責任を見失わず、委託先の力を安定して使うための仕組みである。

## 外部委託は責任放棄ではなく責任分担である

外部委託をすると、作業の一部は外へ出せる。しかし、責任まで全部が外へ移るわけではない。ここを誤解すると、外部委託は必ず崩れる。

Microsoft Learn の `Shared responsibility in the cloud` が 2026年1月12日更新版で明確に示している通り、クラウドでは Microsoft 側へ移る責任がある一方で、どのサービス形態でも customer は data や identities を持ち続け、accounts と access management の責任も retain する。つまり、SaaS やクラウドへ任せても、誰が使えるか、どの権限を持つか、どのデータを守るかは自社の責任に残る。

これはクラウド以外の委託でも同じである。ネットワーク保守を外へ任せても、その変更を承認するのは誰か、どの障害を重大とみなすか、どの時間帯まで対応を求めるか、どこまでの費用を受け入れるかは、自社で決める必要がある。

第6章でまず押さえたいのは、次の区別である。

- 委託先が行う作業
- 自社が持つ判断

作業は任せられる。だが、優先順位、例外判断、重要変更の承認、受け入れ可否、継続可否は自社に残る。この区別が曖昧だと、委託先は「指示待ち」になり、自社は「任せたつもり」で止まる。

## 何を任せて、何を自社に残すか

外部委託がうまくいかない理由の一つは、任せる範囲が曖昧なことだ。曖昧なままでは、委託先は守備範囲を狭く解釈し、自社は広く期待する。そのずれが、障害時と変更時に表面化する。

任せやすいのは、たとえば次のような作業である。

- 定型的な監視
- 定常保守
- 定型設定

- 一次受付
- 一部のキッティングや展開作業

一方で、自社に残すべきものもある。

- 何を優先するか
- 例外を認めるか
- 重要変更を承認するか
- 業務影響をどう評価するか
- どこまで費用をかけるか

たとえば、ネットワーク機器の保守を委託するのはよい。しかし、「設定変更の実作業」は委託しても、「業務影響を伴う変更をいつ行うか」「切り戻すかどうか」の最終判断まで委託先へ流してはいけない。そこは自社の業務都合と責任があるからだ。

また、ヘルプデスクの一次受付を委託することはできる。だが、「どの依頼を標準対応とし、どこから例外申請にするか」というルールは自社で握っておきたい。委託先にすべてを解釈させると、運用ルールそのものが外へ出てしまう。

外部委託をうまく使うとは、仕事を減らすことではなく、どの作業を外へ出し、どの判断を自社に残すかを明確にすることである。

## 委託先に求める最低条件

委託先ごとに長大な評価票を作る必要はない。だが、最低限確認し続ける項目は必要である。NIST の C-SCRM Quick-Start Guide や SP 1326 が示す通り、重要なのは、相手に対して最低限の理解を持ち、要求事項を定義し続けることだ。

第6章の段階では、委託先ごとに次の点が見えていればよい。

- 何を担当する相手か
- どこまでが対応範囲か
- 誰が窓口か
- 平常時と緊急時の連絡方法は何か
- 再委託はあるか
- どのデータや設定に触れるか
- どんな作業報告が返ってくるか

ここで大切なのは、相手の会社案内を読むことではない。自社の運用に必要な情報が取れているかを見ることだ。たとえば、障害時の連絡窓口が営業担当しか分からない状態は危うい。設定変更を依頼した時、作業後に何が報告されるのか分からない状態も危うい。再委託があるのに、どこまで先へ情報が渡るのか見えていない状態も危うい。

委託先管理で必要なのは、完璧な評価ではなく、見えていない部分を減らすことである。

## 連絡経路とエスカレーションを先に決める

障害時に最も無駄が出るのは、原因調査そのものより、「誰に連絡するか」で止まる時間である。平常時は気にならなくても、実際に止まった瞬間には、窓口の曖昧さがそのまま初動遅れになる。

少なくとも、次の四つは先に決めておきたい。

- 一次窓口は誰か
- 緊急時の連絡先は何か

- 社内で判断を引き受ける人は誰か
- 複数ベンダーが絡む時に誰が主導するか

ここで重要なのは、ベンダーごとに窓口を持つだけでは足りないということだ。自社側で、「まず自分が受けるのか」「総務が受けるのか」「社長へどの段階で上げるのか」を決めておく必要がある。

また、複数ベンダーが関わる場面では、押し付け合いが起きやすい。回線事業者、ネットワーク保守業者、クラウドベンダー、社内担当がそれぞれ別だと、「自社設備の問題ではない」「アプリ側の問題ではない」「回線側の調査待ち」となりやすい。そういう時に必要なのは、誰が正しいかをその場で決めることではない。まず誰が事実を集め、誰が暫定判断を出し、誰が次の連絡を回すかを決めることだ。

この章では詳細なインシデント手順までは扱わないが、少なくとも、障害時の連絡経路と承認経路はベンダー管理の一部だと理解しておきたい。

## 外部委託先のアクセス管理と承認

委託先管理で抜けやすいのが、外部委託先のアカウントと権限である。作業のたびに都度渡しているつもりでも、気づくと古いアカウントが残り、共有アカウントが使われ、誰が何をしたのか追えなくなる。

本書の詳しい権限管理は第8章で扱うが、第6章の段階で最低限押さえたい原則は四つある。

- 共用ではなく、できるだけ個人単位で渡す
- 必要最小限の権限にする
- 付与と削除を記録する
- 契約終了時に必ず止める

委託先だからまとめて一つの管理者アカウントでよい、という考え方は危うい。委託先の中でも担当者は変わる。契約終了後にそのアカウントが残れば、外部からの不要な入口になる。さらに、何か起きた時に、誰が行った操作か追えない。

また、ひとり情シスの現場では、「今すぐ見てもらうために、とりあえず広い権限を渡す」が起きやすい。だが、それを後で縮めるのは難しい。だから、最初に広く渡さない方がよい。定常的に必要な権限と、作業時だけ必要な権限は分けて考えたい。

外部委託先のアクセス管理は、相手を信用しないためのものではない。契約、担当者、作業内容が変わっても、入口を見失わないための管理である。

## 定例、レビュー、課題管理を回す

委託先監督というと、重い監査や細かいチェックリストを想像しやすい。だが、個人情報保護委員会の Q&A でも、立入検査そのものが必須とはされていない。個人データの内容や規模に応じて、口頭確認を含む適切な方法でよいとされている。小さな会社では、この考え方が重要である。

つまり、毎回大げさな監査をする必要はない。その代わり、定例と確認項目を持つ。これが現実解になる。

たとえば、主要な委託先とは月一回か隔月で短い定例を持てばよい。確認するのは次のような項目で十分である。

- 先月の障害や問い合わせ
- 未解決課題
- 次回の変更予定
- 契約や更新が近いもの

- 権限やアカウントの見直しが必要なもの

ここで大切なのは、会議を長くすることではない。課題一覧を一つ持ち、期限と担当をはっきりさせることだ。委託先管理が崩れるのは、多くの場合、重大な障害よりも、小さな宿題が誰のものか分からなくなる時である。

また、更新前レビューも有効である。更新時は価格の話だけに寄りやすいが、実際には「この一年で何が詰まったか」「サポートは期待通りだったか」「課題は残ったか」を振り返る良い機会でもある。契約更新を、価格交渉だけの場にしないことが大切だ。

## 再委託と個人データの扱いを見落とさない

委託先管理で特に見えにくいのが、再委託である。自社は A 社へ頼んでいるつもりでも、実際の運用や保守の一部は B 社や C 社が担っていることがある。この時に、「うちは A 社としか契約していないから関係ない」と考えるのは危ない。

個人情報保護委員会のガイドラインでも、必要かつ適切な監督を行っていない場合、再委託先の不適切な取扱いが元の委託元側の問題になり得ると示されている。つまり、再委託は自社の視界の外へ置いてよい話ではない。

第6章で最低限押さえたいのは、次の三点である。

- 再委託があるか
- どの業務やデータが再委託されるか
- 再委託時に事前報告や承認が要るか

加えて、個人データや重要データを扱う委託では、どこに保管されるのか、国外移転があるのか、どのように削除されるのかも確認したい。ここで法制度の詳細解説までは行わないが、「委託した相手の先」まで見ないと説明に詰まる、という現場感だけははっきり持っておくべきである。

## 契約終了時に何を回収するか

ベンダー管理で最後に崩れやすいのは、終了時である。始める時には会議も資料も多いのに、終わる時は急に雑になる。だが、本当は逆である。終了時ほど、データ、権限、接続、手順の回収が必要になる。

最低でも、次の項目は確認したい。

- データ返却
- データ削除確認
- 管理者アカウント削除
- 接続経路の停止
- 手順書や設定情報の引き継ぎ
- 未解決課題や保守履歴の回収

特に危ないのは、「サービスは止めたが、接続は残っている」「契約は終わったが、ベンダーアカウントが残っている」「データを出したつもりだが、必要な形式で取れていない」という状態である。これらは、終了時にしか気づきにくい。

第5章で解約条件やデータ持ち出しを確認したが、第6章ではそれを実際の終了作業へつなぐ。やめる時まで含めて運用である。

## 小さな会社で現実的に回るベンダー管理

ここまで読むと、委託先ごとに重い台帳や監査計画が必要に見えるかもしれない。だが、小さな会社で本当に必要なのは、見える形を一枚持つことだ。

現実的には、次の四つから始めればよい。

- 主要委託先の一覧を作る
- 各委託先の窓口と緊急連絡先を書く
- 外部アカウントを洗い出す
- 終了時の回収項目を決める

主要委託先の一覧には、少なくとも「会社名」「何を任せているか」「社内窓口」「先方窓口」「緊急連絡先」「更新時期」を入れておきたい。これだけでも、障害時と更新時の迷いはかなり減る。

さらに、委託先を重さで分けて考えると回しやすい。たとえば、基幹に近い委託先、日常運用に関わる委託先、単発に近い委託先では、同じ頻度で見なくてよい。重要な相手には定例を持ち、軽い相手は更新前確認だけにする。全部を同じ重さで管理しようとしないことも、ひとり情シスには重要である。

## 通常時の例と、崩れた時の例

通常時の例では、ひとり情シスが主要委託先を一枚にまとめている。ネットワーク保守、回線、クラウド基盤、SaaS　導入支援の四社について、社内窓口、先方窓口、緊急連絡先、任せている範囲、更新時期、外部アカウントを一覧にしている。月一回の短い定例では、未解決課題、今月の変更予定、権限見直し、次回更新を確認する。障害時は、まず社内一次窓口が受け、必要に応じてどの委託先へ上げるかを決める。契約終了時は、データ返却、接続停止、アカウント削除のチェックリストに沿って回収する。結果として、委託先がいても運用の主語は自社のままでいられる。

崩れた時の例では、各委託先の窓口が担当者のメールの中に埋もれている。障害時には、誰へ先に電話すべきかが分からず、回線業者と保守業者が互いに原因を押し付ける。ベンダーには共有の管理者アカウントを渡しており、担当交代も把握していない。契約終了後もアカウントが残り、データ削除の確認もない。委託したはずなのに、見えない不安と後始末がひとり情シスに積み上がる。

この差を生むのは、ベンダーの数でも会社の規模でもない。任せる範囲と見続ける項目を、自社側で持っているかどうかである。

## よくある失敗

第6章で特に避けたい失敗は五つある。

一つ目は、委託した瞬間に自社責任が薄くなると考えることだ。実際には、判断や説明責任は残る。

二つ目は、任せる範囲を曖昧にしたまま運用を始めることだ。曖昧さは、障害時と変更時に一気に問題になる。

三つ目は、窓口と緊急連絡先を整理しないことだ。初動の遅れは、技術不足より連絡不足から起きやすい。

四つ目は、外部アカウントを渡しっぱなしにすることだ。契約終了や担当変更のたびに危険が残る。

五つ目は、再委託と終了時の回収を軽く見ることだ。見えない相手と終わり方の雑さは、後から大きく効く。

## 最低限ここまではやる

第6章の段階では、次の五つができれば十分に前進である。

一つ目。主要委託先を一覧にできること。

二つ目。各委託先の窓口、緊急連絡先、任せている範囲を説明できること。

三つ目。外部委託先のアカウントと権限を把握できること。

四つ目。重要な委託先とは、短くても定例かレビューの場を持てること。

五つ目。契約終了時の回収項目を持っていること。

これだけでも、外部委託はかなり安定する。ベンダー管理とは、相手を細かく縛ることではない。自社が見失ってはいけない責任を、見える形にして持ち続けることである。

**確認したいこと**

- 委託先ごとの責任分界を説明できる
- 障害時の連絡経路を決めている
- 外部委託先のアカウントを把握している
- 再委託の有無を確認している
- 定例やレビューの頻度を決めている
- 契約終了時の回収項目を持っている

## 今日、今週、後でやること

今日やることは三つでよい。まず、主要委託先を五社以内で書き出す。次に、それぞれについて、社内窓口、先方窓口、緊急連絡先、任せている範囲を書き添える。最後に、委託先へ渡しているアカウントや接続経路を洗い出す。

今週やることは、重要な委託先について、短いレビューの場を決めることだ。そこで、未解決課題、今後の変更、権限見直し、更新時期を確認できる形にする。

後で整えることは、再委託確認、終了時チェックリストの整備、主要委託先ごとの運用記録の蓄積である。全部を最初から完璧にしなくてよいが、見えないままにしないことが重要である。

第7章

# 要件定義、導入プロジェクト、変更管理

新しい勤怠システムを入れた。打刻はできる。申請もできる。ベンダーのデモ通りに動いている。だから導入は成功したはずだった。ところが月末になると、締め処理の流れが現場に伝わっていない。CSV の出力仕様が給与計算側と合わない。誰が問い合わせを受けるのかも曖昧で、結局ひとり情シスが全部を拾う。機能は足りていたのに、運用は回らない。

こうした詰まり方は、導入や変更を「実施できるかどうか」だけで進めた時に起きる。技術的に可能でも、目的、影響、戻し方、受け入れ方が曖昧なら、本番後に必ずどこかで詰まる。

第5章では、何を買い、どう契約するかを整理した。第6章では、委託先とどう付き合い続けるかを扱った。第7章では、その次の段階として、要件定義、導入プロジェクト、変更管理を一つの流れで扱う。ここで扱うのは大規模プロジェクト管理の一般論ではない。ひとり情シスが現場で直面する、小さな導入と小さな変更を安全に進めるための型である。

## 課題の整理と導入目的の明確化

導入や変更が崩れる最初の原因は、要件が甘いことではない。その前に、何を解決したいのかが曖昧なことである。

IPA の 重要情報を扱うシステムの要求策定ガイド が、特性評価、問題・リスクと利便性の整理、必要な対策の選定という流れを取っているのは、いきなり対策や機能から入ると判断を誤りやすいからだ。まず整理すべきなのは、今どこで困っていて、誰が困っていて、何を変えたいのかである。

たとえば、勤怠システムを変えたい場合でも、本当の課題は次のどれかで全く意味が違う。

- 打刻漏れが多い
- 承認フローが遅い
- 給与計算への連携が手作業で重い
- テレワーク時の運用に合っていない
- 管理者しか設定変更できず属人化している

課題が違えば、必要な製品も、移行の重さも、導入後の確認点も変わる。にもかかわらず、「今のシステムが古いから変えたい」「新しい方が便利そうだから入れたい」だけで進めると、後で必ず論点が増える。

ここで最初に決めたいのは、次の三つである。

- 何を解決したいのか
- 誰にとっての改善なのか
- 何をもって成功とするのか

この三つが決まると、導入や変更はかなり楽になる。逆にここが曖昧だと、途中で要望が膨らみ、ベンダーとの会話もぶれ、受け入れ確認も曖昧になる。

## 機能要件と非機能要件を分けて考える

目的が見えたら、次は要件である。ここで重要なのは、機能要件と非機能要件を分けて考えることだ。

IPA　の非機能要求グレードが、要求には機能要求と非機能要求があり、非機能要求には要件定義上の課題があると整理しているのは、実務でもこの区別が崩れやすいからである。欲しい機能は言いやすい。だが、止まりにくさ、速さ、ログ、権限、バックアップ、保守性は後回しになりやすい。

機能要件とは、何ができるかである。たとえば、承認フローが二段階で組める、CSV 出力ができる、モバイルから打刻できる、といったものだ。

非機能要件とは、どう支えられるかである。たとえば、次のような点だ。

- どのくらい止まりにくいか
- どのくらいの応答速度が必要か
- どこまでログを残せるか
- 権限をどこまで細かく分けられるか
- バックアップや復元はどうなっているか
- 追加設定や保守を誰がどう行うか

ひとり情シスの現場では、後で苦しくなるのは多くの場合こちらである。機能は足りているのに、管理者権限が雑、操作ログが弱い、保守のたびにベンダー依頼が必要、障害時の切り分けが難しい。こうした詰まり方は、導入時に非機能を言葉にしていなかったことから起きる。

非機能を全部厳密に定義する必要はない。だが、少なくとも重要項目から順に確認する考え方は持ちたい。可用性、権限、ログ、バックアップ、保守性。このあたりは、小さな会社でも先に見ておく価値が高い。

## ステークホルダーと役割分担を決める

導入や変更は、技術担当だけでは終わらない。利用部門、承認者、ベンダー、経営、総務、経理など、関係者が必ずいる。ここを曖昧にすると、途中で「それは聞いていない」「誰が決めるのか分からない」が始まる。

IPA　のアジャイル版モデル契約が、契約前チェックリストの中で、目的・ゴール、ステークホルダー、ビジョン、初期計画、完了基準、品質基準、役割分担の理解を確認しているのは、この認識ずれが失敗の大きな原因だからである。これはアジャイル開発に限らない。SaaS 導入でも設定変更でも同じだ。

第7章で最低限決めたい役割は、次の通りである。

- 誰が利用部門の代表か
- 誰が最終承認するか
- 誰が実施するか
- 誰が本番実施の判断をするか
- 誰が問い合わせを受けるか
- 誰が不具合時の判断を引き受けるか

ひとり情シスが全部を背負いがちなのは分かる。だが、利用部門の要件確認や受け入れ判断まで一人でやると、後で必ず「現場の使い勝手が違う」という話になる。使う人の確認と、承認する人の判断は、できる限り分けたい。

## 導入計画、移行計画、切り戻し計画

本番移行の失敗は、当日発生するように見えて、実際にはその前の計画不足で起きる。ここで分けて考えたいのが、導入計画、移行計画、切り戻し計画である。

導入計画は、全体の段取りである。いつ準備し、いつ試し、いつ本番に入れるかを決める。

移行計画は、何をどの順で移すかである。データ、設定、アカウント、周辺手順のどこまでが対象かを明確にする。

切り戻し計画は、失敗した時にどう戻すかである。ここがないと、本番当日に「このまま続けるしかない」という危険な判断に追い込まれる。

たとえば Wi-Fi 設定変更なら、導入計画は「準備、検証、切り替え、確認」の流れでよい。移行計画では、対象拠点、対象　SSID、対象端末、影響時間帯を整理する。切り戻し計画では、「何分以内に接続不良が一定数を超えたら旧設定へ戻す」といった条件を先に置く。

切り戻しは、単に「元に戻す」と書けばよいわけではない。何を基準に戻すのか、誰が戻す判断をするのか、戻した後に何を確認するのかまで必要である。

また、導入や変更の計画では、連絡も計画に含めるべきである。誰に事前通知し、実施中にどう伝え、失敗時に誰へ上げるか。これがないと、技術的には小さい変更でも、現場から見ると突然止まっただけになる。

## 本番導入と受け入れ確認

本番投入は終わりではない。ここで大事なのは、「入れた」と「使える」を分けることである。

本番前には、少なくとも次を確認したい。

- 対象は正しいか
- 事前バックアップや退避は終わっているか
- 関係者への連絡は済んでいるか

- 切り戻し条件と手順は確認済みか
- 当日の連絡先はそろっているか

本番後には、動作確認だけでなく、受け入れ確認が必要になる。動作確認は、技術的に動くかを見る。受け入れ確認は、業務として使えるかを見る。この二つは違う。

たとえば勤怠システムなら、ログインできる、打刻できる、申請できる、だけでは足りない。締め処理が正しく回るか、承認ルートが運用に合うか、給与計算へ必要なデータが出るか、現場の問い合わせに答えられるかまで見たい。

ここで重要なのが、受け入れ基準を先に置くことである。アジャイル版モデル契約が完了基準や品質基準の明確化を重視しているのも同じ理由だ。終わり方が曖昧だと、「とりあえず本番に入ったから完了」になってしまう。

## 運用引き継ぎまでやって初めて終わる

導入後に特に起きやすい失敗は、「本番開始後の運用」が空白になることだ。ベンダーの作業は終わった。画面も動く。だが、社内では誰が何をするか決まっていない。これでは導入完了とは言えない。

少なくとも、次のものは残したい。

- 手順書
- 問い合わせ窓口
- 標準設定
- 追加変更の流れ
- 不具合時の連絡先
- 保守や監視の範囲

本書では後の章で運用対象ごとの詳細を扱うが、第7章の段階では、「導入したものが日常運用へ渡る形になっているか」を見る。これがないと、ひとり情シスだけが背景を知っていて、現場には何も残らない。

導入をうまく終えるとは、設定作業を終えることではない。後から別の人が見ても回せる状態にすることである。

## 変更管理を日常運用に組み込む

変更管理という言葉を聞くと、大規模システムや厳格な承認プロセスを想像しやすい。だが、NIST SP 800-128 が configuration management を、リスクを最小化しつつ業務上必要な機能を支える discipline としているように、本質はもっと基本的である。変えるなら、何を、なぜ、どこへ、どう戻せるかを持っておくことだ。

ひとり情シスの現場で対象になる変更は多い。

- MFA の強制化
- 権限設計の変更
- Wi-Fi 設定変更
- 端末標準イメージの更新
- 新しい SaaS の全社展開

これらを口頭で進めると、いつ変えたのか、どこまで変えたのか、誰が承認したのか、問題が出たらどう戻すのかが残らない。だから、日常の小変更でも、最低限の変更票を持ちたい。

最低限必要なのは、次の項目である。

- 変更の目的

- 変更対象
- 影響範囲
- 実施日時
- 実施担当
- 戻し方
- 実施後確認

Atlassian の change management 関連資料でも、affected service、planned start、planned end、同時間帯の他変更や進行中インシデントを見てリスクを判断する考え方が示されている。小さな会社でも、これを簡略化して使えばよい。少なくとも、「どこに影響する変更か」「いつやるか」「今ほかに不安定なことはないか」は見たい。

## 標準変更、個別承認変更、緊急変更を分ける

すべての変更を同じ重さで扱う必要はない。むしろ、全部を重くすると続かない。ここで役立つのが、変更の種類を分ける考え方である。

Atlassian の資料でも、standard change は事前承認済みの変更として扱われている。つまり、低リスクで、手順が固まっていて、繰り返し実施する変更は、毎回一から重い承認を取らなくてもよい。

第7章では、次の三つに分けて考えると現実的である。

- 標準変更
  - 手順が固まっている
  - 低リスク
  - 事前承認済み

- 個別承認変更
  - 影響が読み切れない
  - 利用部門確認や承認が必要
- 緊急変更
  - 障害回避や緊急対処のため即時性が高い
  - 事後記録と事後レビューが重要

たとえば、定例の PC 初期設定更新や、手順化された定型設定変更は標準変更にしやすい。一方で、全社 MFA 強制化や基幹に近い設定変更は個別承認変更になる。障害回避のための一時設定変更は緊急変更になる。

この区別があるだけで、現場はかなり回しやすくなる。重く扱うべき変更だけを重くし、それ以外は標準化する。第3章で扱った標準化の原則が、ここでもそのまま効く。

## 小さな会社で現実的に回るやり方

ここまで読むと、プロジェクト管理表や change calendar や会議体が多く必要に見えるかもしれない。だが、小さな会社ではもっと軽くてよい。必要なのは、変更と導入の型を一つ持つことだ。

現実的には、まず次の四つから始めればよい。

- 導入目的を一文で書く
- 変更票に切り戻し欄を入れる
- 受け入れ基準を先に書く
- 変更予定をカレンダーで見える化する

たとえば共有スプレッドシートでもよい。変更一覧に「何を変えるか」「いつやるか」「誰がやるか」「戻し方」「確認結果」を残すだけでも、口頭変更よりはるかに安定する。

また、利用部門との確認は、長い会議でなくてよい。変更前に五分でも、「何が変わるか」「何を確認してほしいか」を共有するだけで、受け入れの質はかなり上がる。

## 通常時の例と、崩れた時の例

通常時の例では、全社 MFA 強制化を進める前に、ひとり情シスが目的を一文で書く。「管理者権限を含むアカウントの不正利用リスクを下げるため、対象者を段階的に MFA 必須へ切り替える」である。その上で、対象者、例外端末、問い合わせ増加、切り戻し条件、当日の連絡先を整理する。受け入れ基準としては、「対象者が翌営業日までにログインできる」「管理者アカウントの MFA が有効」「主要業務で重大な支障がない」を置く。導入後は、問い合わせ対応手順を残し、標準運用へ渡す。結果として、変更が一度のイベントで終わらず、日常運用に接続される。

崩れた時の例では、ベンダーの勧めで新しい勤怠システムを急いで入れる。比較は機能中心で、締め処理、CSV 出力、権限、問い合わせ運用は深く見ていない。移行当日はデータ投入に追われ、うまくいかなければその場で考えるしかない。受け入れ基準もなく、利用部門は「一応動くなら開始しましょう」となる。本番後に細かな不整合と問い合わせが噴き出し、ひとり情シスは導入後の火消しを続けることになる。

この差を生むのは、プロジェクトの規模ではない。変える前に、目的、影響、戻し方、受け入れ方を言葉にしたかどうかである。

## よくある失敗

第7章で特に避けたい失敗は五つある。

一つ目は、課題整理をせずに製品や設定の話から始めることだ。これでは要件がすぐぶれる。

二つ目は、機能要件だけを見て非機能要件を後回しにすることだ。後で運用が苦しくなるのはたいていこちらである。

三つ目は、移行計画と切り戻し計画を分けて考えないことだ。失敗時に戻せない変更は危うい。

四つ目は、受け入れ確認を「動いたかどうか」だけで済ませることだ。業務として使えるかを見る必要がある。

五つ目は、日常変更を記録せず、口頭やチャットだけで進めることだ。後で原因も責任も追えなくなる。

## 最低限ここまではやる

第7章の段階では、次の五つができれば十分に前進である。

一つ目。導入目的を一文で説明できること。

二つ目。機能要件と非機能要件を分けて書けること。

三つ目。移行計画と切り戻し計画を持てること。

四つ目。受け入れ基準を先に決めていること。

五つ目。日常の変更について、最低限の記録を残していること。

これだけでも、導入と変更はかなり安定する。ひとり情シスの仕事を守るとは、全部を慎重に遅くすることではない。変える時に、必要な確認だけは先に済ませることである。

**確認したいこと**

- 導入目的を一文で説明できる
- 機能要件と非機能要件を分けている
- 移行計画と切り戻し計画を持っている
- 受け入れ基準を決めている
- 導入後の運用引き継ぎを考えている
- 日常変更を記録している

## 今日、今週、後でやること

今日やることは三つでよい。まず、今進んでいる変更や導入を一件選び、その目的を一文で書く。次に、その案件について、非機能要件を三つだけ書き足す。最後に、切り戻し条件と切り戻し手順を一行でも入れる。

今週やることは、変更票や導入メモのひな型を作ることだ。そこに、目的、対象、影響、実施日時、戻し方、確認結果の欄を入れる。これだけで、場当たり変更はかなり減る。

後で整えることは、標準変更の整理、変更予定のカレンダー化、導入後ナレッジの蓄積である。全部を最初から制度化する必要はないが、口頭変更のままにしないことが重要である。

第III部

# 日常運用の基盤

人、端末、SaaS、データを崩れにくく回す。

第8章〜第16章

第8章

# アカウント、認証、権限管理

退職者のメールは止めた。社用 PC も返却された。だからアカウント対応は終わったはずだった。ところが数週間後、共有フォルダのアクセス履歴を見ると、その人が所属していたプロジェクトの権限がまだ残っている。さらに調べると、プロジェクト管理ツール、経費精算、ファイル共有にも同じような残り方があった。加えて、設定変更は共有の `admin@` アカウントで行われていたため、誰がどこを変えたのかも追えない。

こうした問題は、アカウント管理を「作る」「消す」という作業で捉えた時に起きる。実際には、それだけでは足りない。誰として扱うか。どう本人確認するか。どこまで入れるか。後で追えるか。この四つがそろって初めて、アカウント管理が回り始める。

第7章では、導入や変更を安全に進めるための型を整理した。第8章では、その変更のたびに必ず論点になるアカウント、認証、権限管理を扱う。ここで扱うのは、製品ごとの設定手順ではない。小さな会社でも崩れにくい、IAM の最小原則である。

## ID、認証、権限、監査を分けて捉える

まず言葉を分けておきたい。ここが混ざると、議論がすぐ曖昧になる。

ID とは、誰として扱うかである。社員本人の個人アカウントなのか、管理者用の別アカウントなのか、連携用のサービスアカウントなのか。まず主体を切り分ける必要がある。

認証とは、その　ID　の本人性をどう確かめるかである。パスワードだけなのか、MFA を使うのか、SSO でまとめるのか、セキュリティキーやパスキーまで使うのか。ここは「ログイン方法」の話である。

権限とは、認証が通ったあとにどこまで入れるかである。閲覧だけか、編集までか、管理者か、監査ログまで見られるか。認証と権限は別物である。本人確認ができても、何でもできてよいわけではない。

監査証跡とは、誰がいつ何をしたかを後で追えるようにすることである。管理者権限の付与、グループ変更、MFA の再設定、共有設定の変更、緊急用アカウントの利用などが追えなければ、問題が起きた時に原因も責任も分からない。

この四つを一体で見ると、アカウント管理の見え方が変わる。たとえば退職者対応は、メール停止だけでは終わらない。本人の　ID　を無効化し、関連する認証手段を止め、残っている権限を外し、実施記録を残して初めて終わる。第9章ではその流れを詳しく扱うが、第8章では、その前提となる考え方を固めたい。

## パスワード、MFA、SSO を一体で考える

認証の話になると、今でも「強いパスワードを定期変更する」という発想に寄りがちだ。だが、NIST の現在の整理はかなり違う。長く、固有で、管理しやすいパスワードを使い、根拠なく頻繁な変更を強制しない。むしろ、長いパスフレーズ、パスワードマネージャー、コピーアンドペーストの許容、SSO　のような利用者負荷を減らす設計を重視している。

実務で大事なのは二つである。一つ目は、同じパスワードを複数サービスで使い回さないこと。二つ目は、パスワードだけで守ろうとしないことだ。IPA　も、不正ログイン被害の増加に対して、ID とパスワードだけでは弱く、パスキーや多

要素認証の導入を勧めている。漏えい、推測、リスト型攻撃、フィッシング。こうした現実を考えると、特に管理者権限や重要データへ触るアカウントでは、MFA は前提と考えた方がよい。

ただし、MFA を入れれば何でもよいわけではない。NIST が AAL2 でフィッシング耐性のある選択肢を提供すべきとしているように、認証アプリ、セキュリティキー、パスキーのような方式は、SMS だけに頼るより堅い場面が多い。ひとり情シスの現場でも、最低でも管理者アカウントから優先的に、可能ならフィッシング耐性の高い方式へ寄せたい。

ここで忘れやすいのが、復旧手段である。MFA は強いが、端末故障、機種変更、紛失、回線障害で自社管理者が入れなくなる事故も起きる。強くすることと、復旧できることは両立させなければならない。この論点は後で改めて扱う。

SSO も、便利だから入れるだけではもったいない。NIST が federation によって認証負荷を下げられると示している通り、SSO は利用者の手間を減らす。そのうえで、ひとり情シスにとってはもっと重要な価値がある。認証ポリシー、MFA、無効化、ログ、棚卸しを一か所へ寄せやすくなることだ。SaaS が増えるほど、アプリごとに個別ログインを持つ運用は崩れやすい。全てを一度に SSO 化できなくても、全社利用 SaaS、管理者権限がある SaaS、退職時に漏らしたくない SaaS から順に寄せる価値は高い。

ただし、SSO に寄せるほど、IdP 側の管理者権限と復旧手段はより重要になる。認証を集中させるとは、守るべき中枢を一つに絞ることでもあるからだ。

## 権限設計は個人直付けではなく役割で行う

権限管理が崩れる一番の原因は、個人ごとの例外設定が増え続けることだ。入社時に少し足す。異動時に一部だけ残す。急ぎの依頼で直接付与する。これを繰り返すと、半年後には誰がなぜその権限を持っているのか分からなくなる。

ここで必要なのが、役割で考えることだ。CIS も Microsoft も、最小権限を個人への我慢ではなく、役割、範囲、期間で整理する方向を示している。つまり、「営業担当ならここまで」「経理担当ならここまで」「情シス管理者でも日常利用はここまで」という標準形を先に作る。

現場では、次の三つで考えると分かりやすい。

- 役割
  - 一般利用者
  - 部門責任者
  - 経理や人事など機微情報を扱う担当者
  - 情シス管理者
  - 外部委託先
- 範囲
  - どのアプリか
  - どのフォルダか
  - どの組織、どのプロジェクトか
- 期間
  - 恒久か
  - プロジェクト期間限定か
  - 緊急対応中だけか

この考え方にすると、権限付与は人ごとの職人芸ではなく、グループやロールへの所属管理へ変わる。異動時も、個別権限を一つずつ見直すより、所属グループを変える方が速く、漏れにくい。第3章で扱った標準化が、ここでもそのまま効く。

最小権限という言葉は、厳しさだけで理解すると誤る。これは事故が起きた時の被害面積を減らす設計であり、同時に、従業員のプライバシーや業務分掌を守る設計でもある。Google　が管理者権限を必要最小限へ絞るよう整理しているのも、そのためである。見られなくてよい情報は見られない方がよい。

また、権限は永続でなくてよい。システムが対応していれば、申請時だけの昇格、一定時間で失効する付与、承認付きの一時権限を使うと崩れにくい。そこまで高度な機能がなくても、「この権限は今月末で見直す」と期限を持たせるだけで、放置される権限は減る。

## 管理者権限を日常利用と分離する

通常利用アカウントと管理者アカウントを分ける。これは第8章の中でも特に重要な原則である。

管理者権限を持つアカウントで、メールを読み、Web を見て、日常の事務作業をするのは危うい。万一そのセッションや端末が侵害された時、被害は一気に広がる。Microsoft も Google も、日常利用と高権限利用を分離することを明確に勧めている。ひとり情シスであっても、この原則は変わらない。

理想は、次のように分けることだ。

- 通常利用アカウント
  - メール
  - チャット
  - 文書作成
  - 一般的な業務利用
- 管理者アカウント
  - 設定変更

- 権限付与
- 監査ログ確認
- 管理画面操作

加えて、最上位権限は少人数へ絞りたい。Microsoft は Global Administrator を 5 人未満に抑えることを勧めている。製品名は違っても考え方は同じで、何でもできる権限を広く配らない方がよい。

ひとり情シスの現場では、「自分一人しかいないから分けても意味がない」と思いやすい。だが、意味はある。一つは、通常作業中の事故面積を減らせること。もう一つは、監査証跡が見やすくなることだ。通常ログインと管理操作が混ざらないだけでも、後で追いやすくなる。

もし利用しているサービスが対応しているなら、常時管理者にするのではなく、必要時だけ昇格できる仕組みが望ましい。PIM のような機能がなくても、管理者権限を持つ人数を絞り、付与や変更の記録を残すだけでかなり違う。

## 緊急用アカウントと復旧手段を持つ

認証を強くするほど、設計を誤った時のロックアウトも重くなる。ここは実務で非常に重要だが、見落とされやすい。

Google は 2SV 保護アカウントの復旧にあたり、予備のセキュリティキー、バックアップコード、追加の super administrator（最上位管理者）を勧めている。Microsoft も緊急用アカウントの考え方を明確に示している。要するに、最後の復旧手段まで同じ前提で縛ってはいけないということだ。

小さな会社でも、次のものは持ちたい。

- 緊急用の管理者アカウント

- 予備の認証手段
  - 予備セキュリティキー
  - バックアップコード
  - 回復用連絡先
- 利用時の通知
- 保管場所と取り出し手順

緊急用アカウントは、日常利用しない。個人のメール用途にも使わない。平時は封印し、使ったら必ず記録し、なぜ使ったかを残す。利用時に経営者や責任者へ通知が飛ぶ運用にできるとさらによい。

重要なのは、MFA 強制や条件付きアクセスの変更、SSO 切り替えのような大きな認証変更を行う時に、緊急用アカウントまで同じ変更で巻き込まないことである。最後の一枚の鍵まで同時に締めると、自社が中へ入れなくなる。

また、復旧手段は持っているだけでは足りない。年に一度でもよいので、予備キーの所在、バックアップコードの更新、緊急用アカウントのサインイン可否を確認したい。災害備品と同じで、必要な時に使えなければ意味がない。

## 共有アカウントとサービスアカウントを例外管理する

原則は個人アカウントである。誰が何をしたかを追うには、それしかない。

共有アカウントは、一見便利だ。代表メール、店舗端末、共用　PC、機器管理画面など、事情がある場面もある。だが、共有アカウントは監査証跡を壊しやすく、退職や異動の影響範囲も見えにくい。Google　が共有管理者アカウントを避けるよう勧めているのも当然である。

やむを得ず共有アカウントを残すなら、少なくとも次を持たせたい。

- 用途
- 所有者
- 利用者の範囲
- 認証情報の保管場所
- MFA をどう扱うか
- パスワード変更時期
- 廃止できる見込み

特に管理者権限を共有アカウントで持つのは避けたい。共有の代表メールが必要でも、管理者権限は個人アカウントへ委任した方がよい場面が多い。

サービスアカウントも放置されやすい。API　　連携、バックアップ、ジョブ実行、スクリプト、監視。便利な一方で、人の異動と無関係に生き残るため、見直しから漏れやすい。CIS もサービスアカウントの棚卸しを独立論点にしている。

サービスアカウントには、少なくとも次を残したい。

- 何のためのアカウントか
- どのシステムが使うか
- 所有者は誰か
- 秘密情報はどこに保管されているか
- 権限は何か
- 次の見直し日はいつか

人が日常利用するアカウントを、そのまま自動処理へ流用するのは避けたい。逆に、サービスアカウントで人が手作業ログインするのも避けたい。主体が違うからである。

## 定期棚卸しと監査証跡を回す

権限は、放っておくと必ず増える。異動、兼務、臨時対応、外部委託、導入時の暫定設定。だから棚卸しが必要になる。

ここで重要なのは、棚卸しを単なる一覧確認にしないことだ。Microsoft の access review が、対象、レビュー担当、周期、未回答時の扱い、結果反映まで設計させているのは、見るだけでは元に戻らないからである。

棚卸しでは、少なくとも次を決めたい。

- 何を見直すか
  - 管理者権限
  - グループ所属
  - 共有フォルダ権限
  - 外部委託先権限
  - サービスアカウント
- 誰が判断するか
  - システム所有者
  - 部門責任者
  - 情シス
- いつやるか
  - 四半期
  - 半年
  - 年次

- 未回答ならどうするか
- 不要と判断した権限をいつ外すか
- 記録をどこへ残すか

最初から厳密な仕組みは要らない。だが、周期は決めたい。実務では、管理者権限、外部委託先、機微データアクセスは四半期ごと、一般的な権限は半年ごとから始めるのが現実的である。CIS は休眠アカウント無効化の目安として 45 日を示しているが、実務では休職や季節業務もあるため、自社基準として何日で確認対象にするかを先に決めておく方が回しやすい。

監査証跡として特に見たいのは、次のようなものだ。

- 管理者権限の付与と剥奪
- グループ所属変更
- MFA の再設定
- 共有設定変更
- 緊急用アカウントの利用
- 失敗したログインや異常なサインイン

共有管理者アカウントを使っていると、ここが全部ぼやける。誰がやったかではなく、「その共有 ID で誰かがやった」までしか分からない。これは運用上かなり弱い。

## 小さな会社で現実的に回るやり方

ここまで読むと、専用の IAM 製品や高度な統制基盤が必要に見えるかもしれない。だが、小さな会社ではもっと単純でよい。まずは台帳と周期を持つことから始めればよい。

最低限、一覧にしたい項目は次の通りである。

- 氏名または所有者
- アカウント名
- 種別
  - 個人
  - 管理者
  - 共有
  - サービス
- 対象システム
- 所属グループまたは役割
- MFA 状態
- 復旧手段の有無
- 作成日
- 最終見直し日
- 終了予定日

この一覧だけでも、かなり見え方が変わる。特に、管理者アカウント、共有アカウント、サービスアカウントを分けて見られるだけで、危ない場所が浮く。

運用としては、次の形が始めやすい。

- 管理者権限一覧は四半期ごとに見直す
- 一般権限一覧は半年ごとに見直す
- 共有アカウントは用途と所有者を毎回確認する
- 休眠アカウントは月次で抽出する
- 変更管理の記録と権限変更をひも付ける

全ての SaaS をすぐ SSO 化できなくても、SSO 非対応アプリを一覧へ明示し、個別に無効化が必要だと分かる状態にしておけば、退職時や棚卸し時に漏れにくくなる。

## 通常時の例と、崩れた時の例

通常時の例では、新しいプロジェクト管理ツールを導入する際に、ひとり情シスが最初に個人アカウント原則を置く。ログインは　SSO　へ寄せ、部門ごとのグループで基本権限を配る。管理者操作は通常利用アカウントとは別の管理者アカウントで行い、管理者権限は四半期ごとに見直す。一般権限は半年ごとに棚卸しし、外部委託先は契約更新のたびに確認する。MFA を強制する前に、予備キーとバックアップコード、緊急用管理者アカウントも整える。異動が起きても、グループを変えるだけで大半の調整が終わる。誰が設定を変えたかも追える。

崩れた時の例では、各 SaaS に個別ログインがあり、権限は依頼のたびに直接付与している。管理者権限は共有の　`admin@`　で回し、複数人が同じ認証情報を知っている。MFA は慌てて有効化したが、予備手段は準備していない。ある日、役員のスマートフォン故障をきっかけに管理者が入れなくなり、さらに過去の退職者権限も残っていたことが判明する。調べようにも、共有アカウント運用のため誰が何を変えたか追えない。問題は個別の設定ミスではなく、アカウント管理を統制として見ていなかったことにある。

## よくある失敗

第8章で特に避けたい失敗は五つある。

一つ目は、アカウント管理を作成削除だけの話にすることだ。これでは権限残りと監査不全が起きやすい。

二つ目は、パスワードを厳しくすれば十分だと思うことだ。今は、固有性、長さ、MFA、復旧手段の方が重要である。

三つ目は、管理者権限を通常利用と混ぜることだ。事故面積が広がり、操作追跡もしにくくなる。

四つ目は、共有アカウントやサービスアカウントを例外扱いせず放置することだ。たいてい後で棚卸しから漏れる。

五つ目は、棚卸しを予定だけで終わらせ、結果反映と証跡を残さないことだ。見たが直していない、が一番危うい。

## 最低限ここまではやる

第8章の段階では、次の五つができれば十分に前進である。

一つ目。個人、管理者、共有、サービスのアカウント種別を分けて把握していること。

二つ目。管理者アカウントを通常利用アカウントと分けていること。

三つ目。管理者アカウントと重要 SaaS に MFA を設定し、復旧手段も持っていること。

四つ目。共有アカウントとサービスアカウントに所有者と用途があること。

五つ目。管理者権限は四半期、一般権限は半年を目安に棚卸しする予定があること。

これだけでも、アカウント管理はかなり安定する。ひとり情シスが全部を覚えている状態から、後で見返しても回る状態へ一歩進めるからだ。

**確認したいこと**

- ID、認証、権限、監査を分けて説明できる
- 管理者アカウントは通常用と分かれている
- 最上位権限の人数を把握している
- MFA と復旧手段をセットで整えている
- 共有アカウントとサービスアカウントに所有者がいる
- 定期棚卸しの予定と証跡がある

## 今日、今週、後でやること

今日やることは三つでよい。まず、管理者権限を持つアカウントを全部一覧にする。次に、その中で通常利用と管理者利用が混ざっているものへ印を付ける。最後に、共有アカウントとサービスアカウントの所有者欄を埋める。

今週やることは、MFA 未設定の管理者アカウントをなくし、予備の復旧手段を一つ用意することだ。同時に、次回の棚卸し日をカレンダーへ入れる。予定が入っていない棚卸しは、だいたい実施されない。

後で整えることは、SSO 対象の拡大、期限付き権限付与、監査ログ確認の定例化である。全部を一度にやる必要はないが、個人、管理者、共有、サービスの区別だけは曖昧にしない方がよい。

第9章

# 入社、異動、退職とアカウント運用フロー

退職者のメールは止めた。だから対応は終わったと思っていた。ところが午後になると、営業から「引き継ぐはずの取引先メールが見られない」と連絡が来る。総務からは「共有フォルダの中身も必要らしい」と言われる。さらに調べると、その退職者はスマートフォンからまだ一部のアプリに入れ、委託先向けの共有リンクも生きていた。止めたつもりだったのに、仕事もアクセスも中途半端に残っていた。

こうした詰まり方は、入社、異動、退職を単発作業で捉えた時に起きる。実際には、人が動くたびに、アクセス、データ、端末、記録の状態も変えなければならない。

第8章では、ID、認証、権限、監査を一体で捉える原則を整理した。第9章では、その原則を人のライフサイクルへ落とし込む。ここで扱うのは、人事総務の一般論ではない。ひとり情シスが漏らしやすい、入社、異動、退職、休職、委託終了時の IT 運用を、崩れにくい順番で整理する章である。

## 入社、異動、退職は統制イベントである

人の状態が変わると、会社から見たアクセス状態も変わる。だから、入社、異動、退職は人事イベントであると同時に、統制イベントでもある。

入社では、まだ何にも入れない人を、必要な範囲だけ入れる状態へ変える。異動では、前の役割で持っていた権限を見直し、新しい役割へ合わせ直す。退職では、入れていたものを止め、残すべきデータを移し、不要なものを閉じる。

この三つを別々の作業として見ると、たとえば次のような抜けが起きる。

- 入社時
  - アカウントは作ったがライセンスがない
  - MFA が未設定
  - 端末が届いていない
- 異動時
  - 新しい権限だけ足した
  - 旧部署の共有フォルダが残る
  - 承認権限が二重に残る
- 退職時
  - メール停止だけで終えた
  - セッションが残る
  - データ移管前に削除した

第8章で見たように、アカウント管理とは、誰として扱い、どう本人確認し、どこまで入れ、後で追えるかを崩さず回すことだった。第9章では、その考え方を時間軸に沿って扱う。人に紐づく状態は固定ではない。変化に合わせて更新する必要がある。

## 入社時に最初から整えるべきもの

入社対応で最も避けたいのは、「とりあえず使える」状態のまま始めることだ。初日に使えないのも困るが、急いだ結果として危うい状態を作るのも困る。

Microsoft 365 も Google Workspace も、新規ユーザー作成時に、アカウント名、初期パスワード、ライセンス、所属、権限、連絡先をまとめて整える前提になっている。つまり、入社時に必要なのはメールアドレス発行だけではない。

最低限、次のものは入社時にそろえたい。

- 個人アカウント
- 所属部署や組織情報
- 標準ライセンス
- 標準権限
- MFA 設定
- 端末準備とのひも付け
- 本人への引き渡し方法

ここで重要なのは、個別例外から始めないことだ。第8章で整理したように、権限は個人直付けより、役割とグループで持たせた方が崩れにくい。入社時は、その人に一つずつ権限を付けるより、「営業の標準」「経理の標準」「情シスの標準」といった型へ乗せる方がよい。

また、初回パスワードの渡し方も軽視しない方がよい。Google の新規ユーザー作成でも、二次連絡先へ情報を送る流れが示されている。Microsoft 365 でも、資格情報の印刷や安全な共有が前提になっている。つまり、「チャットに平文で投げる」が標準になってはいけない。

入社時は、準備の順番も大事である。現実的には次の流れが回しやすい。

1. 人事または依頼元から確定情報を受ける
2. アカウントと所属情報を作る
3. ライセンスと標準権限を付ける
4. 端末準備とひも付ける

5. MFA を有効にする
6. 初期情報を安全に渡す
7. 初日確認をする

Google Workspace では、新規アカウントが各サービスへ反映されるまで最大 24 時間かかることがある。つまり、「当日朝に依頼が来てもすぐ全部使える」とは限らない。ひとり情シスの現場では、入社連絡の締切を持った方がよい。今日入る人の依頼が今日の朝に来る運用は、事故の元である。

## 異動時は旧権限の剥奪を先に考える

異動時に起きやすい失敗は、新しい権限を足すことに意識が向き、古い権限を外し忘れることである。

異動は退職ほど緊張感が出にくい。本人は社内に残るので、つい「使えなくなると困るから、とりあえず今のままにしておこう」になりやすい。だが、その積み重ねが権限肥大化を生む。

異動時に見直したいのは、少なくとも次のものだ。

- 所属グループ
- 共有フォルダ権限
- メーリングリスト
- 承認権限
- 管理者権限
- 外部サービスのロール

考え方としては、まず旧所属の標準権限を外し、そのうえで新所属の標準権限を付ける。兼務の場合だけ、両方を残す理由を明示する。これが基本である。

ここで重要なのは、「この人は前から使っていたから」という理由だけで残さないことだ。前部署の共有フォルダ、旧案件の管理画面、前任者として持っていた承認権限。こうしたものは本人の利便性には見えても、会社から見ると不要アクセスかもしれない。

異動時にアカウントを作り直す必要は、普通はない。むしろ、同じ人である以上、履歴やデータを保ちつつ、所属と権限を変える方が自然である。削除と再作成は、履歴の分断や引き継ぎ漏れを生みやすい。

異動対応で現実的に回しやすいのは、依頼票へ次の欄を持つことだ。

- 旧所属
- 新所属
- 外す権限
- 残す権限
- 新たに付ける権限
- 実施日
- 確認者

この一行があるだけで、「足したが外していない」をかなり減らせる。

## 休職、長期不在、委託終了は削除ではなく一時遮断を基本にする

人の状態が変わったからといって、いつも削除が正しいわけではない。休職、産休育休、長期出張、調査中、委託先の一時停止。こうした場面では、一時遮断の方が適切なことが多い。

Google Workspace の suspend は、アクセスを止めるがデータは消さない構造になっている。Microsoft 側でも、block sign-in やセッション失効を使って、まずアクセスだけ止められる。これは実務上かなり重要だ。

一時遮断が向くのは、次のようなケースである。

- 休職中
- 長期不在
- 不正利用の疑いがあり調査中
- 委託先の作業が一時停止
- 退職確定前だが先にアクセスを止めたい

削除してしまうと、戻す時に手間が増え、履歴も切れやすい。逆に停止なら、必要時に復元しやすい。

ただし、一時遮断でも放置してよいわけではない。端末回収、MFA トークンの扱い、共有リンクの確認、委託先の VPN やゲスト権限の停止などは別途必要になる。アクセス遮断と周辺整理は分けて考えたい。

委託先対応で特に危ういのは、「社員ではないから人事イベントに乗らない」ことだ。実際には、契約終了は退職に準じるイベントである。委託先の個人アカウント、ゲストアカウント、共有フォルダ、チケットシステム、VPN、チャット参加権限は、終了日に合わせて止めなければならない。

## 退職時の初動は、遮断、失効、切り離しである

退職時に最初にやるべきことは削除ではない。遮断である。

Microsoft　の元従業員対応ガイドでも、最初の工程はサインイン防止であり、その後にデータ保存や端末ワイプへ進む。Google　でも、退職後のデータ保護として、端末ワイプ、復旧用連絡先削除、パスワード変更が示されている。つまり、初動でやるべきことは一貫している。

退職時の初動は、次の順番で考えるとよい。

1. サインインを止める
2. パスワードを変更する
3. セッションやトークンを失効させる
4. 回復手段を切り離す
5. モバイル端末や BYOD 上の業務データを消す

ここで「セッションやトークンを失効させる」を独立させるのが大事だ。パスワードを変えても、すでに発行済みのセッションやトークンが生きていれば、すぐには切れないことがある。Microsoft Entra でも refresh token とセッションクッキー失効の機能が独立しているのは、そのためである。

また、復旧用メールアドレスや電話番号が本人の私物にひも付いたままだと、後から回復経路として使える可能性がある。退職時は、認証だけでなく回復経路も切る必要がある。

即日退職や不正疑いの場面では、この初動がさらに重要になる。この時は、データ引き継ぎやライセンス処理より先に、まず入れない状態を作る。後続作業はそのあとでよい。

## メールとファイルの引き継ぎを分けて考える

退職対応でよくある誤解は、「データ引き継ぎ」と一言でまとめてしまうことだ。実際には、メールとファイルでは考え方が違う。

メールでは、主に次の選択肢がある。

- 一定期間の転送
- 共有メールボックス化
- PST などで書き出して保管
- 必要な人へ閲覧権を渡す

ファイルでは、主に次の選択肢になる。

- OneDrive や個人ストレージへ管理者が一時アクセスする
- 別担当者へアクセス権を渡す
- 所有権を移す
- 共有ライブラリや共有ドライブへ移す

この違いを意識しないと、「見られるようにしたつもりだが所有権は元のまま」「転送はしたが過去メールは見られない」といったズレが起きる。

Google Drive の所有権移管では、共有権限そのものは変わらず、有効なアカウントへの移管が前提とされている。つまり、誰が所有者かと、誰が見られるかは別論点である。Microsoft 365 でも、OneDrive へのアクセス付与と Outlook データの引き継ぎは別工程として整理されている。第9章では、この違いを曖昧にしない方がよい。

また、共有メールボックスやメール転送を使う場合、Microsoft 365 では元アカウントがアンカーとして必要になる。つまり、転送や共有化を設定したのに元アカウントをすぐ削除すると成り立たないことがある。順番を間違えないことが重要である。

実務では、退職対応票に次の欄を持つと漏れにくい。

- メール転送の要否

- 共有メールボックス化の要否
- 過去メール保存の要否
- OneDrive や Drive の引き継ぎ先
- 所有権移管の要否
- 実施期限

## 停止、ライセンス削除、アーカイブ、削除を使い分ける

第9章で特に整理したいのが、状態の違いである。ここが混ざると、不要な削除や無駄な課金が起きる。

まず、停止である。これはアクセスを止めるが、データは保持する。休職、調査中、退職直後の初動に向いている。

次に、ライセンス削除である。これは課金対象サービスの利用権を外す操作であり、必ずしもアカウント削除ではない。Microsoft 365 のガイドでも、ライセンス削除とユーザー削除は別工程になっている。

次に、アーカイブである。Google Workspace では archived user にすることで、アクセスを止めつつデータを保持し、アクティブライセンスを再利用できる。保持とコストの両立が必要な組織では有力な選択肢になる。

最後に、削除である。これは最終工程である。2026年3月時点では、Microsoft では通常 30 日、Google Workspace では 20 日の復元可能期間があるが、その期間を過ぎると戻せない。しかも、何が保持され、何が残らないかはサービスごとに違う。

考え方としては、次のように使い分けると分かりやすい。

- すぐ止めたい
  - 停止
  - セッション失効
- まだ戻る可能性がある
  - 停止
  - 端末回収
- 保持したいが課金は抑えたい
  - アーカイブが使えるなら検討
- 引き継ぎと保存が終わった
  - 削除判断

削除を急ぐと、後で困る。特に、復元可能期間とデータ保持条件を把握せずに削除すると、「必要なファイルがあった」「法務確認でメールが要る」となった時に詰まる。

## 外部委託先、ゲスト、短期利用者も同じ型で扱う

社員の入退社は意識されやすいが、外部委託先やゲストは見落とされやすい。だが、権限が残れば同じように危うい。

特に確認したいのは、次の対象だ。

- ゲストユーザー
- 委託先個人アカウント
- 一時的な共有フォルダ権限

- プロジェクトごとのチャネル参加
- VPN やリモート接続
- チケットシステムや監視画面の閲覧権

委託先の終了日は、人事システムに乗らないことが多い。だからこそ、契約終了日とアカウント停止日を結び付ける必要がある。第6章で見たように、外部委託は責任分担であり、終了時の回収まで含めて設計しなければならない。

短期利用者も同じである。短いから大丈夫ではなく、短いからこそ終了確認を忘れやすい。終了日を持ち、終わったら止める。この単純な運用が重要である。

## 小さな会社で現実的に回るやり方

JML を厳密なワークフローシステムで回せる会社ばかりではない。小さな会社では、まず 1 枚のチェックリストでも十分効果がある。

最低限、次の項目は一枚にまとめたい。

- イベント種別
  - 入社
  - 異動
  - 休職
  - 退職
  - 委託終了
- 対象者
- 実施日
- 人事連絡日
- アカウント

- ライセンス
- 権限
- MFA
- セッション失効
- 端末
- メール引き継ぎ
- ファイル引き継ぎ
- 削除またはアーカイブ判断
- 実施記録

運用としては、次のように分けると回しやすい。

- 事前にやること
  - 入社準備
  - 通常退職準備
  - 異動予定の確認
- 当日にやること
  - サインイン遮断
  - 権限変更
  - 端末回収
- 後でやること
  - データ移管
  - ライセンス再利用
  - 削除またはアーカイブ
  - 記録保存

特に退職時は、当日やることと後でやることを分けた方がよい。当日は止めることに集中し、引き継ぎや保存はその後に順序立てて進める方が安全である。

## 通常時の例と、崩れた時の例

通常時の例では、新入社員の入社日が一週間前に共有される。ひとり情シスは、アカウント、ライセンス、部署グループ、端末、MFA を事前に整え、初日朝に安全な方法で初期情報を渡す。異動者については、旧部署の権限一覧を確認し、残すものと外すものを先に決める。退職者については、最終出社時刻に合わせてサインイン遮断、パスワード変更、セッション失効を実施し、その後にメール転送、共有メールボックス化、OneDrive や Drive の引き継ぎを進める。削除やアーカイブは、移管確認後に判断する。結果として、人の出入りが起きても慌てない。

崩れた時の例では、入社連絡は当日朝に来る。アカウントだけ急いで作り、ライセンスも MFA も後回しになる。異動では、旧部署権限が残ったまま新部署権限を足す。退職時には、メール停止だけ先にやり、後で必要になったファイルが見られず、復元期間内に慌てて戻すことになる。委託先アカウントは契約終了後も共有フォルダへ入り続ける。問題は忙しさだけではない。人の状態変更を、統制イベントとして設計していなかったことにある。

## よくある失敗

第9章で特に避けたい失敗は五つある。

一つ目は、入社時にアカウントだけ作って終わることだ。ライセンス、MFA、端末、標準権限が抜ける。

二つ目は、異動時に足すことばかり考え、外すことを後回しにすることだ。権限肥大化の典型である。

三つ目は、退職時に削除を急ぐことだ。まず遮断し、その後に引き継ぎと保存を行うべきである。

四つ目は、メールとファイルの引き継ぎを同じだと思うことだ。閲覧権、所有権、保持期間は別々に見る必要がある。

五つ目は、外部委託先やゲストを本流のチェックリストから外すことだ。見落としやすいが、危険度は低くない。

## 最低限ここまではやる

第9章の段階では、次の五つができれば十分に前進である。

一つ目。入社、異動、退職、委託終了を一枚のチェックリストで管理していること。

二つ目。異動時に旧権限を外す欄があること。

三つ目。退職時の初動が、削除ではなく、遮断、失効、回復手段切り離しになっていること。

四つ目。メール引き継ぎとファイル引き継ぎを分けて考えていること。

五つ目。停止、アーカイブ、削除の違いと、復元可能期間を把握していること。

これだけでも、人の出入りに伴う事故はかなり減る。ひとり情シスの負荷を減らすとは、全部を自動化することではない。抜けやすい順番を先に整えることである。

### 確認したいこと

- 入社、異動、退職の標準手順がある
- 異動時に旧権限を外す欄がある
- 退職時の初動が削除ではなく遮断になっている
- メールとファイルの引き継ぎ方法を分けている
- 停止、アーカイブ、削除の使い分けを説明できる
- 復元可能期間と証跡保存を把握している

## 今日、今週、後でやること

今日やることは三つでよい。まず、入社、異動、退職、委託終了を同じ様式で扱えるチェックリストを作る。次に、その様式へ「旧権限を外す欄」と「セッション失効欄」を足す。最後に、主要サービスごとの復元可能期間を一覧にする。

今週やることは、直近の異動者か退職者を一件選び、実際にそのチェックリストへ当てはめてみることだ。そこで詰まった欄が、自社の抜けである。

後で整えることは、人事連絡の締切、端末回収手順との接続、外部委託先終了の自動通知である。全部を最初からシステム化する必要はないが、人が動くたびにゼロから考える運用はやめた方がよい。

第10章

# PC、スマホ、端末、資産管理

退職者のアカウントは止めた。だから対応は終わったと思っていた。ところが数日後、棚に置かれた返却 PC を次の入社者へ回そうとして手が止まる。前の利用者のローカルファイルは消えたのか。暗号化は有効だったのか。管理画面ではまだその人の端末として見えている。貸与スマートフォンも一台行方が曖昧で、誰が持っているかすぐには言えない。

こうした詰まり方は、端末管理を「買って配る作業」と見た時に起きる。実際には、PC　もスマートフォンも、配布して終わりではない。標準化して、管理下へ載せて、利用中の状態を追い、紛失時に止め、返ってきたら消して、再び配るか廃棄するかを決める必要がある。

第9章では、人のライフサイクルに合わせてアカウントと権限を変える運用を見た。第10章では、その裏側にある端末のライフサイクルを扱う。ここで扱うのは、キッティングの小技ではない。ひとり情シスが、少ない人数でも端末を崩れにくく回すための基本設計である。

## 端末管理はライフサイクル管理である

端末管理というと、購入、初期設定、貸与の三つだけを思い浮かべやすい。だが実際には、端末は次の流れで動く。

1. 調達する
2. 管理対象として登録する
3. 標準構成を適用する

4. 利用者へ配布する
5. 利用中の状態を監視し、更新する
6. 紛失、盗難、故障に対応する
7. 回収する
8. 再配備するか、退役するかを決める

このどこかが曖昧だと、端末はすぐに管理の外へ落ちる。たとえば、購入時に台帳へ載っていない端末は、誰が持っているか分からなくなる。配布時に管理下へ入っていない端末は、更新や遠隔操作ができない。返却後の消去手順がない端末は、前任者のデータを抱えたまま次の人へ渡りかねない。

CIS Controls では、組織が保有、接続、管理する資産を特定し、承認されていない資産を把握することを基礎統制として置いている。つまり、第10章で最初に持つべき感覚は、「今ある端末が全部言えるか」である。

## 先に標準構成を決める

端末管理で最も崩れやすいのは、毎回その場で設定を決める運用である。急ぎの入社者が出るたびに、昨日の設定を思い出しながらアプリを入れ、権限を付け、ブラウザを調整する。これでは、同じ機種でも中身がそろわない。

先に決めるべきなのは、機種より標準構成である。最低限、次の項目は持っておきたい。

- OS とバージョン方針
- 必須アプリ
- 業務ブラウザと拡張
- 画面ロック
- ディスク暗号化

- ウイルス対策や EDR の有無
- ローカル管理者権限の扱い
- 更新の適用方針

ここで重要なのは、完璧な標準を目指しすぎないことだ。ひとり情シスの現場では、最初から部署別に細かく分けすぎると維持できない。まずは、全社共通の標準 PC、必要なら少数の例外構成、というくらいに絞った方がよい。

ローカル管理者権限も、端末管理を崩しやすい論点である。何でもできる状態にしておくと、利用者が困った時に自分で入れられて楽に見える。だが、その結果、勝手に入ったアプリ、止まった更新、原因不明の設定変更が積み上がる。第10章では、「管理者権限を配るかどうか」ではなく、「標準から外れる変更をどう管理するか」で考えたい。

## 台帳は台数確認ではなく状態把握に使う

会計上の資産台帳があっても、運用として十分とは限らない。資産番号と購入日だけ分かっても、今の運用判断には足りないからである。

運用台帳として最低限ほしいのは、次の項目である。

- 資産番号
- 端末種別
- メーカーと型番
- シリアル番号
- 利用者
- 所有区分
- 管理方式

- 最終接続日
- 現在状態
- 保管場所または返却予定

ここでいう所有区分は、会社所有か、レンタルか、私物利用かである。管理方式は、MDM 管理下か、個別設定か、未管理かである。現在状態は、利用中、予備機、故障中、返却待ち、退役予定などである。

この項目を持っていると、単に何台あるかだけでなく、次の判断ができる。

- 誰の端末が長く接続していないか
- 返却予定なのにまだ回収できていない端末はどれか
- 会社所有なのに管理下へ入っていない端末はないか
- 故障交換時に代替機を出せるか

Google Workspace にも、非アクティブな会社所有端末を把握する機能がある。Microsoft Intune にも、長期間接続していない端末記録を cleanup 対象として整理する考え方がある。つまり、公式資料でも「使われていない端末や古い記録を見直す」ことは標準運用として扱われている。

## 配布と初期設定を個人技にしない

標準構成を決めても、それを端末へ安定して入れられなければ意味がない。ここで価値を持つのが、自動登録や MDM である。

Microsoft の Windows Autopilot は、初回配布から再利用や廃棄までを含むデバイスライフサイクルの一部として説明されている。Apple の Automated Device Enrollment も、組織所有端末を初回アクティベーションから MDM 管理下へ確実

に乗せる仕組みである。重要なのは製品名ではなく考え方だ。つまり、端末は「受け取った人が自由に設定するもの」ではなく、「会社の標準設定が自動で入るもの」として扱う方が崩れにくい。

配布時に持ちたい工程は、次の通りである。

1. 端末を運用台帳へ登録する
2. 管理対象として登録する
3. 標準構成を適用する
4. 利用者をひも付ける
5. 貸与記録を残す
6. 初回サインインと利用開始を確認する

この順番の中で、貸与記録を軽く見ない方がよい。誰にいつ渡したか、いつ返却予定か、付属品は何か。この記録がないと、故障時や退職時に「その端末は誰のものだったか」が曖昧になる。

また、予備機をどう持つかも決めておきたい。全台ぎりぎりで回していると、故障時に復旧が遅れる。小さな会社でも、標準構成済みの予備機を少数持つだけで、障害対応はかなり安定する。

## 利用中管理では更新、権限、休眠端末を見る

配布後の端末管理は、気合いでは続かない。見る項目を絞る必要がある。ひとり情シスが最低限追いたいのは、更新、権限、状態の三つである。

まず、更新である。OS　と主要アプリの更新が止まると、脆弱性対応も不具合修正も止まる。標準構成を作る時点で、いつ、どこまで自動適用するかを決めた方がよい。

次に、権限である。端末ローカルの管理者権限が広く残っていると、標準から外れた変更が増え、問題切り分けが難しくなる。必要な例外は記録し、ずっと例外のまま放置しないことが大切である。

最後に、状態である。特に見たいのは、しばらく接続していない端末である。使っていないだけに見えても、実際には次のような可能性がある。

- 返却されたが台帳更新が漏れている
- 退職者がまだ持っている
- 出張用として貸し出されたまま戻っていない
- 故障しているが放置されている
- 管理から外れている

休眠端末は、単なる無駄ではなく、管理抜けの兆候である。月次で一度確認し、利用者不明、長期未接続、退役予定のものを洗い直すだけでも効果がある。

## スマホと BYOD は管理境界を先に決める

PC　と比べると、スマートフォンは私物利用と業務利用が混ざりやすい。ここで曖昧にすると、事故時にもめる。

会社所有スマホであれば、比較的考えやすい。会社管理下へ登録し、必要な設定を配り、紛失時に lock や wipe を実行できる状態を目指せばよい。

難しいのは BYOD である。NIST の BYOD 関連資料でも、組織データ保護と利用者プライバシー保護の両立が課題として明示されている。つまり、BYOD　を採るなら、最初に次の境界を決める必要がある。

- 会社データだけ消せるのか
- 端末全体を消せるのか

- どこまで設定を強制するのか
- どこまで端末状態を見えるのか
- 利用者へどう説明するのか

Google Workspace でも、管理レベルによって、会社データだけの wipe と端末全体の wipe が分かれている。つまり、BYOD では「何かあったら全部消す」が当然にはできない。ここを曖昧にしたまま運用すると、事故時に実行できると思っていた操作ができずに詰まる。

ひとり情シスとしては、BYOD　を標準にしない方が運用は安定する。もし採るなら、対象業務、対象アプリ、消去範囲、利用者同意、退職時の処理を先に決めてから始めるべきである。

## 紛失、盗難、故障時の初動を決めておく

端末事故で最も危ういのは、その場で考える運用である。紛失の連絡を受けてから、誰が何を止めるのかを考え始めると遅い。

最低限、初動は次のように固定しておきたい。

1. 連絡を受ける
2. 端末種別と所有区分を確認する
3. 対象アカウントのリスクを判断する
4. 必要に応じてサインインを止める
5. 端末側の lock または wipe を実施する
6. 回線停止や SIM 停止が必要なら行う
7. 記録を残す
8. 代替機を手配する

ここで大事なのは、アカウント対応と端末対応を分けることである。第9章で見たように、アカウント停止やセッション失効は重要である。だが、それだけでは端末内に残るローカルデータやオフラインファイルまでは扱えない。Microsoft Intune や Google Workspace の資料が遠隔 wipe を独立機能として持っているのは、そのためである。

故障時も同じである。壊れたから交換する、で終わりではない。返ってきた端末のデータ消去、管理状態の更新、代替機の貸与記録が必要になる。故障をきっかけに台帳が崩れることは多い。

## 回収、再配備、退役を分けて考える

返却された端末は、ただの中古端末ではない。状態を判定して、次の行き先を決める必要がある。

まず回収である。ここでは、端末本体だけでなく、充電器、SIM、周辺機器、貸与記録を確認する。

次に状態確認である。故障の有無、管理下に残っているか、暗号化状態、返却理由を見て、再利用可能かを判断する。

次にデータ消去である。ここを飛ばしてはいけない。Apple の公式資料でも、消去は再設定や再配備前の独立工程として扱われている。Microsoft Intune の wipe も、退役や再利用時に使うことが前提になっている。つまり、「初期化したつもり」ではなく、「標準手順に沿って消去した」と言える状態が必要である。

そのうえで、再配備するなら標準構成を再適用し、新しい利用者へひも付ける。退役するなら、廃棄、返却、売却、保管のどれかを決め、台帳状態を変える。

ここで特に注意したいのが、管理解除である。Apple Business Manager の device release は、組織所有でなくなった端末だけに行うべきで、しかも不可逆である。つまり、初期化と管理解除は同じではない。まだ会社所有で再配備する可能性がある端末を、勢いで管理から外してはいけない。

回収後の工程は、最低でも次の四つに分けたい。

- 回収した
- 消去した
- 再配備待ちにした
- 退役または廃棄した

この状態が分かれていれば、「返ってきたけれど次に渡してよいのか」が曖昧になりにくい。

## 小さな会社で現実的に回るやり方

すべての端末を高度な管理基盤で統一できる会社ばかりではない。小さな会社では、まず運用を細くしすぎないことが重要である。

現実的には、次のくらいから始めると回しやすい。

- 標準機種を PC とスマホで少数に絞る
- 標準構成票を 1 枚作る
- 端末台帳へ利用者、最終接続日、状態欄を入れる
- 配布時と返却時に必ず記録を残す
- 月に一度、休眠端末を確認する
- 紛失時の初動手順を 1 枚にする
- 退職時チェックリストへ端末回収欄を入れる

第9章の JML フローとつなげると、端末管理はかなり安定する。入社時は、誰にどの標準端末を渡すかを決める。異動時は、端末を変えるか、設定だけ変えるかを判断する。退職時は、返却、消去、再配備待ちまでを一続きで処理する。人と端末を別々に動かさないことが大切である。

## 通常時の例と、崩れた時の例

通常時の例では、新入社員の入社連絡が一週間前に来る。ひとり情シスは、標準 PC を一台確保し、運用台帳へ登録し、会社管理下へ入れ、標準構成を適用しておく。初日に利用者へ貸与し、貸与記録を残す。退職時は、最終出社日に端末を回収し、状態確認後に消去し、再配備待ちへ移す。スマートフォン紛失時は、あらかじめ決めた手順に沿って、アカウント確認、端末 lock、必要に応じた wipe、回線停止、代替機手配まで進める。結果として、誰が何を持っているか、返ってきた端末が次に使えるかをすぐ判断できる。

崩れた時の例では、端末は調達のたびに機種が違い、設定内容も担当者の記憶頼みである。貸与記録が曖昧なので、退職者の返却 PC がどれかすぐに分からない。スマートフォンを紛失しても、会社所有か私物利用かが即答できず、どこまで消せるか判断できない。返ってきた端末は、とりあえず初期化したつもりで次の人へ回される。結果として、配布、回収、再配備のたびに毎回調べ直しになり、事故が起きた時には止めるべきものが見つからない。

## 最低限ここまではやる

第10章の内容をすべて一度に整えなくてもよい。だが、次の四つは早めに持ちたい。

1. 標準端末構成を決める
2. 利用者と状態が見える台帳を持つ

3. 紛失時の初動を固定する
4. 回収後の消去と再配備の手順を分ける

この四つがあるだけで、端末管理はかなり崩れにくくなる。端末管理とは、物を配ることではない。会社管理下へ置き続け、必要時に止め、戻し、消し、また配れる状態を保つことである。

## 今日、今週、後でやること

今日やることは、端末台帳を開き、利用者欄と状態欄が埋まっているか確認することである。空欄が多いなら、そこが最初の詰まりどころである。

今週やることは、標準 PC と標準スマホの構成票を 1 枚ずつ作ることである。完璧でなくてよい。OS、必須アプリ、暗号化、画面ロック、管理方法が書かれていれば十分に価値がある。

後でやることは、紛失時フローと返却後フローを、JML の流れとつなげて見直すことである。人が動いた時に端末も動く。この一本線が引けると、第10章の内容は実務へ落ちる。

第11章

# ネットワーク、拠点、テレワーク

「ネットが遅いです」と言われても、それだけでは何も分からない。Wi-Fi が不安定なのか、回線自体が落ちているのか、特定の会議室だけなのか、在宅勤務者だけなのか。ところが実際の現場では、社内 Wi-Fi とゲスト Wi-Fi が同じ設定のまま、VPN の入口も複数あり、どの回線が主回線でどこへ連絡すればよいかも一枚で説明できないことがある。これでは、障害が起きた時に最初の一手が決まらない。

ネットワークは、専門家だけが触る謎の箱ではない。もちろん細かな設定は専門性が高い。だが、ひとり情シスが押さえるべき本質は別にある。回線がどこから入り、どこで分かれ、社内用と外部向けがどう分かれ、在宅勤務者はどこから入るのか。この構造を説明できることが、運用責任の出発点である。

第10章では、端末を会社管理下へ置き続ける運用を見た。第11章では、その端末をつなぐ基盤を扱う。ここで扱うのは、ルーターの CLI 操作ではない。オフィス、拠点、Wi-Fi、テレワーク接続を、少人数でも崩れにくく回すための考え方である。

## ネットワーク管理の最初の仕事は、図で説明できるようにすること

ネットワークが難しく感じる最大の理由は、見えないまま動いているからである。だから最初にやるべきことは、高度な最適化ではなく、構成を見えるようにすることだ。

CIS Control 12 では、ネットワーク構成図や関連文書を維持し、少なくとも年 1 回、または大きな変更時に見直すべきとしている。つまり、構成図は立派な設計資料ではなく、日常運用の必須資料である。

最初の構成図は細かくなくてよい。最低限、次が分かれば十分に役立つ。

- 回線事業者と回線の入口
- ルーターやファイアウォール
- 主なスイッチ
- Wi-Fi アクセスポイント
- 社内用ネットワーク
- ゲスト用ネットワーク
- VPN や在宅勤務の入口
- 重要な接続先

この図があると、障害や変更のたびに効く。たとえば、「在宅勤務者だけつながらない」なら VPN 側を見る。「ゲストだけつながらない」なら SSID か DHCP 側を見る。「全社でインターネットが出られない」なら回線や境界機器を見る。図がないと、この切り分けがいつも頭の中だけで行われ、属人化する。

構成図と一緒に持ちたいのが、連絡先の一覧である。回線事業者、保守業者、機器ベンダー、社内窓口。この一覧がないと、障害時にまず電話番号を探すことになる。ネットワーク障害時は、調査力より先に、連絡の速さが効くことが多い。

## 社内、ゲスト、機器用を分ける

ネットワーク設計で最初に持ちたい原則は、用途の違うものを混ぜないことである。

NIST の WLAN ガイドでは、内部利用と外部利用でセキュリティプロファイルが違うなら、論理的に分離した WLAN を用意すべきだとしている。ゲスト用 WLAN からは、原則として内部ネットワークを通らず、必要最小限だけに絞る考え方である。これは大企業向けの厳格論ではない。小さな会社ほど効果が大きい。

少なくとも、次の区分は分けて考えたい。

- 社員が業務で使うネットワーク
- 来客や一時利用者向けのゲスト Wi-Fi
- 会議室機器やサイネージなどの共用機器
- プリンタ、複合機、IoT 機器

全部を一つにすると、問題が起きた時の影響範囲が広がる。来客用 Wi-Fi の設定ミスが社内端末へ波及し、古い会議室機器が同じセグメントにいるせいで切り分けが複雑になる。逆に分かれていれば、「どこまで止まっているか」を狭めやすい。

ここで必要なのは、難しい設計より、最低限の役割分離である。小さな会社なら、まずは社内用とゲスト用を分けるだけでもよい。余裕があれば、共用機器や IoT を別にする。いきなり複雑な VLAN 設計を作って維持できなくなるより、少数の役割分離を守る方が現実的である。

## Wi-Fi は SSID とパスワードだけでは設計したことにならない

Wi-Fi は手軽に見えるぶん、雑に運用されやすい。だが、無線は有線以上に境界を意識した方がよい。

NIST SP 800-153 では、WLAN は設計、導入、保守、監視のライフサイクル全体で考えるべきとしている。つまり、「パスワードを設定したから終わり」ではない。

Wi-Fi で最低限押さえたいのは、次の論点である。

- SSID の役割分け
- 暗号化方式
- 認証方式
- 接続先ネットワーク
- クライアント同士の分離
- アクセスポイントやルーターの更新

暗号化方式は、CIS でも secure protocols の例として 802.1X や WPA2 Enterprise 以上が挙げられている。Cisco Meraki の無線セキュリティ資料でも、WEP は安全ではなく、WPA3 や WPA2 Enterprise がより適した選択肢として整理されている。ひとり情シスとしては、少なくとも古い方式を放置しないこと、どの SSID がどの認証方式を使っているか説明できることが必要である。

また、ゲストや BYOD 向け SSID では、client isolation の考え方が重要になる。Meraki の資料でも、guest や BYOD 向け SSID で、無線クライアント同士の通信を抑える機能が有効だと整理されている。これは高度な追加機能ではなく、「同じ Wi-Fi にいる他人の端末へむやみに届かせない」ための基本である。

Wi-Fi 運用で地味に効くのは、SSID を増やしすぎないことだ。部署ごと、部屋ごと、用途ごとに増やすと、誰がどこにつなぐべきか分からなくなる。基本は少数の役割で分ける。社内用、ゲスト用、必要なら機器用。このくらいから始めた方が崩れにくい。

## テレワークは、入口とアクセス範囲を決める

在宅勤務や出張先からの接続は、「どこでも仕事できて便利」で終わらせると危うい。NIST SP 800-46 は、外部環境は敵対的である前提で、遠隔接続を設計すべきとしている。つまり、自宅回線もカフェ Wi-Fi も、社内ネットワークと同じ前提では扱えない。

ここで最初に決めたいのは、誰が何からどこへ入れるかである。NIST は、端末種別ごとにアクセス範囲を階層化する考え方を示している。たとえば、会社管理 PC は広く許可し、BYOD PC は限定し、スマホは web メール程度に抑える、といった考え方である。

この発想は、ひとり情シスにとってかなり有効である。全部を同じように許可しようとすると、運用がすぐ崩れるからだ。最低限、次の区分は持ちたい。

- 会社管理 PC からの接続
- 私物 PC からの接続
- スマートフォンからの接続
- 外部委託先からの接続

それぞれについて、どの接続方式を認めるかを決める。CIS Control 12 でも、遠隔端末は企業管理 VPN と AAA 基盤へ接続してから業務資源へ入る考え方を示している。つまり、「必要なら好きな遠隔ツールでつないでよい」ではなく、会社が決めた入口を通す方が運用しやすい。

ここで注意したいのが、遠隔接続方式の乱立である。VPN、リモートデスクトップ、各種遠隔操作ツール、SaaS 側の個別ログインが場当たり的に増えると、誰がどこから入れるか分からなくなる。CISA の遠隔接続ソフト保護ガイド

も、正規ツールが攻撃に悪用される前提で、承認済みツールの利用と監視を求めている。第11章では、テレワークを理由に入口を増やしすぎない方がよいと書きたい。

また、在宅勤務ルールは VPN 手順だけでは足りない。CISA は自宅 Wi-Fi について、ルーター更新、強いパスワード、WPA3 または WPA2 AES、不要なリモート管理の停止、接続端末の確認を勧めている。会社が自宅ネットワークを全面管理できなくても、「最低限ここは守ってほしい」という基準は持てる。

## 拠点回線とバックアップ経路も運用対象にする

ネットワーク機器は意識されやすいが、回線は見落とされやすい。だが、実際には主回線、ONU、ルーター、ISP 障害、工事、オフィス移転の方が業務へ大きく効くことがある。

ここで持ちたいのは、単一障害点の視点である。次のどれが止まると、どこまで止まるかを把握したい。

- 主回線
- 予備回線
- ONU やモデム
- 境界ルーターやファイアウォール
- 拠点間接続
- DNS や認証など接続に必要な周辺サービス

Cisco Meraki の MultiWAN 資料でも、複数 ISP やセルラー回線を組み合わせて可用性を上げるユースケースが整理されている。さらに、バックアップ回線有効化のような変更は、通信断を伴うためメンテナンス時間帯に行うことが推奨されている。これは製品固有の話ではなく、回線切替そのものが変更管理対象だという意味で重要である。

小さな会社でも、最低限の暫定手段は持っておきたい。たとえば次のようなものだ。

- スマートフォンのテザリング
- モバイルルーター
- 予備回線
- 一時的に別拠点へ業務を寄せる手順

ここで大切なのは、完全無停止を目指すことではない。止まった時に、どこまで続けられるかを先に決めることである。受注だけは続けたいのか、メールだけ維持できればよいのか、在宅へ切り替えるのか。この判断があると、障害時の迷いが減る。

## 障害時は、まずどこまで止まっているかを分ける

ネットワーク障害対応で最も重要なのは、最初の切り分けである。ここを外すと、社内で迷い、外部連絡も遅れる。

一次切り分けでは、まず次の順に分けるとよい。

1. 全社か、一部の人だけか
2. 有線も無線も両方か、Wi-Fi だけか
3. 社内だけか、在宅勤務者だけか

4. インターネット全体か、特定の社内サービスだけか
5. 直前に設定変更や工事があったか

この五つだけでも、行き先はかなり絞れる。たとえば、全社かつ有線も無線も両方なら、回線か境界機器を見る。Wi-Fi だけなら SSID、AP、無線側の DHCP や認証を見る。在宅勤務者だけなら VPN、認証、相手側の自宅回線や端末設定を見る。

重要なのは、「ネットワークが悪い」という一言で全部をまとめないことだ。問題の層を分けるだけで、連絡先も変わる。ISP へ連絡するのか、保守業者か、社内設定変更の切り戻しか、利用者の自宅 Wi-Fi 見直しか。この順番が決まっていると、ひとり情シスでも慌てにくい。

障害票やメモには、最低限次を残したい。

- 発生時刻
- 影響範囲
- 有線か無線か
- 拠点名
- 直前変更の有無
- 一次確認結果
- 外部連絡先
- 暫定対応

後で振り返る時にも、この情報が効く。同じ会議室だけ繰り返し切れる、同じ時間帯に VPN が遅くなる、といった再発傾向が見えてくるからである。

## 小さな会社で現実的に回るやり方

すべての拠点に高度な冗長化やゼロトラスト基盤を入れられる会社ばかりではない。小さな会社では、まず少ないルールを固定する方が効果が高い。

現実的には、次のくらいから始めると回しやすい。

- 構成図を 1 枚持つ
- 社内用とゲスト用の SSID を分ける
- Wi-Fi の暗号化方式と認証方式を確認する
- 在宅勤務の接続方式を一つか二つに絞る
- 回線事業者と保守業者の連絡先を一覧化する
- ISP 障害時の暫定手段を決める
- 障害一次切り分け票を 1 枚にする

ここで意図的に避けたいのは、中途半端に複雑な構成である。VLAN も SSID も VPN も乱立しているのに、誰も全体像を説明できない状態が一番危うい。少数の役割で分け、少数の接続方式に絞り、その代わり構成図と連絡先を最新に保つ方が、ひとり情シスには向いている。

## 通常時の例と、崩れた時の例

通常時の例では、ひとり情シスがネットワーク構成図を一枚持っている。主回線、予備回線、ルーター、主要スイッチ、社内 Wi-Fi、ゲスト Wi-Fi、VPN の入口が載っている。社内用とゲスト用の SSID は分かれ、ゲスト側は内部へ届かない。在宅勤務は会社指定の接続方式へ統一され、私物端末は閲覧系だけに限定さ

れている。ISP　障害時には、まず主回線障害か内部機器障害かを切り分け、必要ならテザリングや予備回線へ切り替える。結果として、「どこで止まっているか」と「次に誰へ連絡するか」がすぐ決まる。

崩れた時の例では、Wi-Fi の SSID は増え続け、どれが社内用でどれが旧設定か分からない。来客も社員も同じ無線へ入り、会議室機器も同じネットワークにぶら下がる。在宅勤務は人によって使う接続方式が違い、VPN も遠隔操作ツールも増えている。回線障害が起きても、ISP　　の契約番号も保守窓口もすぐ出てこない。結果として、障害そのものより「どこから調べるか」で時間を失う。

## 最低限ここまではやる

第11章の内容をすべて一度に整えなくてもよい。だが、次の四つは早めに持ちたい。

1. 現在の構成図を 1 枚作る
2. 社内用とゲスト用を分ける
3. テレワークの標準入口を決める
4. 障害一次切り分け票と外部連絡先一覧を作る

この四つがあるだけで、ネットワークはかなり管理しやすくなる。ネットワーク管理とは、全部を自分で設定することではない。どこがどこにつながり、誰がどこまで使え、どこで止まるかを説明できる状態を保つことである。

## 今日、今週、後でやること

今日やることは、紙でもスライドでもよいので、今のネットワーク構成を一枚に書き出すことである。正確でなくてもよい。回線、境界機器、Wi-Fi、VPN の入口が見えるだけで価値がある。

今週やることは、使っている SSID とその接続先を棚卸しすることである。社内用なのか、ゲスト用なのか、誰向けなのか、内部へ届くのか。この整理だけでも、かなりの曖昧さが消える。

後でやることは、在宅勤務ルールと回線障害時の暫定運用を一枚にまとめることである。ネットワークは止まることがある。だからこそ、止まった時の動き方を先に決めておくことが、ひとり情シスの実務では大きい。

第12章

# SaaS、クラウド、業務アプリ管理

現場が「便利だから」と言って契約した SaaS は一つだけだった。ところが半年後に棚卸ししてみると、その SaaS は Google アカウントでログインし、Zapier とつながり、ファイルは別のストレージへ送られ、通知はチャットへ流れ、外部委託先とも共有されていた。契約書の枚数は増えていないのに、データの出口は増えていた。

SaaS 管理が難しいのはここにある。問題は、サービス本体だけではない。誰が使っているか、どの連携アプリが権限を持っているか、外部共有がどこまで開いているか、解約する時に何を持ち出せるか。こうした周辺まで見ないと、SaaS はすぐに統制の外へ広がる。

第5章では、調達と契約を設計として扱った。第8章では、認証と権限の原則を見た。第12章では、その先にある実運用を扱う。ここで扱うのは、製品比較ではない。導入後の SaaS とクラウドサービスを、少人数でも崩れにくく管理するための考え方である。

## SaaS 管理は、利用実態を見える化するところから始まる

SaaS を契約一覧だけで見ていると、実態を見失う。契約していることと、使われていることは同じではないからだ。

たとえば、次のようなずれはよく起きる。

- 契約はあるが、ほとんど使われていない
- 契約した担当者が退職し、管理者が曖昧になっている

- 全社利用のつもりが、一部署だけで別運用になっている
- 無料枠や個人契約のまま使われている

だから第12章で最初に持ちたいのは、契約台帳だけでなく、運用台帳である。最低限、次は見えるようにしたい。

- サービス名
- 用途
- 主担当部門
- 管理者
- 利用者の範囲
- 取り扱いデータ
- 認証方式
- 外部共有の有無
- 主要な連携アプリ
- 更新日
- export 可否

ここで重要なのは、「何を契約したか」より「今どう使われているか」を書くことだ。たとえば、営業支援 SaaS 一つを取っても、顧客データが入るのか、取引先へ共有するのか、他サービスへ API 連携しているのかで、管理の重さは変わる。

## 野良 SaaS を見つけ、承認済みの型へ寄せる

現場が先に使い始める SaaS を、完全にゼロにするのは難しい。だから必要なのは、見つけて、分類して、統制下へ寄せることである。

Microsoft Defender for Cloud Apps は、導入初期のステップとして cloud discovery を置き、どの cloud apps が、誰に、どの端末で使われているかの可視化を重視している。これはとても実務的である。野良 SaaS の問題は、規程違反そのものより、見えていないことだからだ。

ひとり情シスとしては、まずサービスを三つに分けると運用しやすい。

- 承認済み
- 要審査
- 禁止

承認済みは、会社として使ってよいもの。要審査は、業務上必要かもしれないが、データ種別、認証、共有、export 可否を見てから判断するもの。禁止は、機密データを扱えない、権限管理が粗い、連携制御が弱いなど、会社方針として避けるものだ。

ここで大切なのは、見つかった時の扱いである。現場を責めるだけでは、別の場所でまた増える。まず用途を確認し、承認済みサービスで代替できるなら寄せる。代替できないなら、要審査として統制条件を満たせるかを見る。この順番の方が実務では回る。

## OAuth、API、ノーコード連携は別の入口として管理する

SaaS　管理で特に見落としやすいのが、連携アプリである。ユーザーは「ログインしただけ」「便利機能を有効にしただけ」のつもりでも、実際にはデータへの新しい入口が増えていることがある。

Google Workspace の app access control は、Google-owned、Internal、Third-party の三種類のアプリを区別し、Trusted、Specific Google data、Limited、Blocked という水準で制御できる。さらに、Google 側は requested scopes や verified status、ユーザー数まで見えるようにしている。つまり、Google も「本体サービス」と「それに届くアプリ」を分けて管理する前提で作っている。

Microsoft Entra でも同じである。ユーザー同意をどう許可するかを制御でき、verified publisher に絞ることが推奨されている。さらに、許可済みの権限は後から review して revoke できる。つまり、OAuth 同意は一回きりの儀式ではなく、継続的に見直すべき統制点である。

ここで管理対象に含めたいのは、次のものだ。

- ユーザーが自分で同意した OAuth アプリ
- 管理者が全社同意したアプリ
- API キーを使う連携
- Zapier のようなノーコード連携
- service account や app account

特に、Zapier のような自動化は便利だが、複数サービスの間にデータ経路を作る。どのデータが、どの条件で、どこへ送られるのかを説明できない状態で増やすと危うい。

Microsoft は consent phishing という形で、悪意あるアプリへ同意させる攻撃を明示している。つまり、連携アプリは利便性の問題であると同時に、攻撃経路の問題でもある。だから第12章では、少なくとも次を見たい。

- 誰が許可したか
- 何の権限を要求しているか
- verified か

- 今も使われているか
- 不要なら revoke できるか

## 外部共有は、機能ではなく公開境界である

SaaS　やクラウドアプリの共有設定は、単なる便利機能ではない。どこまでを社外へ見せるかという公開境界である。

Google Workspace の Drive では、外部共有はデータ流出リスクを伴うとして、warning や link sharing の制限、組織単位でのオンオフ、共有専用 shared drive の活用などが示されている。Microsoft SharePoint でも、Anyone、New and existing guests、Existing guests、Only people in your organization というように、外部共有の水準を複数段階で持っている。

ここで重要なのは、共有を「オンかオフか」で終わらせないことだ。実務では、少なくとも次を分けて考えたい。

- 誰が共有できるか
- 誰と共有できるか
- 認証が必要か
- リンク転送だけで見られるか
- どの領域で共有を許すか

Microsoft は、特定の security group だけに外部共有権限を持たせる方法も案内している。Authenticated guests only を選べば、外部共有しても相手の認証が必要になる。これはとても重要だ。全員が、どこでも、誰にでも共有できる状態にしておく必要はない。

また、既定値を放置しない方がよい。SharePoint では site type によって default sharing setting が違う。OneDrive が `Anyone` になりうる構成もある。つまり、「特に変えていないから安全だろう」は危うい。

第12章では、共有を個人の善意へ委ねず、共有可能者、共有可能範囲、認証要件を会社側で先に決める考え方を書きたい。

## クラウドの責任分界を理解する

クラウドでは、提供者が多くの運用を担う。だが、それは責任が消えることと同じではない。

Microsoft Azure は、すべてのクラウド展開形態で、データと ID は顧客が持つ責任だと明示している。Google Cloud も、顧客は自社の要件に応じて必要な security controls を自ら設定しなければならないとしている。AWS も、顧客責任は選ぶサービス形態に応じて変わるが、データ、権限、設定は顧客側で考える必要があると整理している。

この共通点から、第12章で押さえたいのは次である。

- データは誰の責任か
- ID とアクセス制御は誰が持つか
- 設定ミスは誰が防ぐか
- ログはどこまで取れるか
- export や削除は誰が実行するか

SaaS では基盤運用の負担は減る。だが、共有設定、管理者権限、連携アプリ、データ分類、退職時の無効化までは自社責任であることが多い。ここを曖昧にすると、「ベンダーに聞けば何とかなる」と思っていたのに、実際には自社設定の問題だった、ということが起きる。

## 利用終了時の出口を先に決める

SaaS　は入れる時より、終わらせる時に詰まりやすい。請求停止だけで終わらないからである。

利用終了時には、少なくとも次を見たい。

- どのデータを出すか
- どこへ保管するか
- どの形式で出せるか
- どれくらい時間がかかるか
- 共有や連携を先に止める必要があるか
- 旧データをいつ削除するか

Google Workspace の Data Export tool では、export は最短でも 48 時間後、通常 72 時間、場合によっては 14 日かかる。Google 提供バケットに出した場合は 60 日で自動削除される。つまり、「退職日までに急いで全部ダウンロードしておこう」のような発想は危うい。出口にも準備が要る。

この考え方は Google Workspace に限らない。どの SaaS でも、導入前に次を確認した方がよい。

- export 機能があるか
- 監査ログを出せるか
- 共有データの所有権を移せるか
- 解約後の保持期間はどうなるか
- API 連携先をどう止めるか

第12章では、「導入前に出口を見る」という癖を持たせたい。出口を考えずに始めると、移行や解約のたびに人が張り付く。

## 小さな会社で現実的に回るやり方

すべての SaaS を高度な CASB や SaaS セキュリティ製品で管理できる会社ばかりではない。小さな会社では、まず少数のルールを固定する方が効く。

現実的には、次のくらいから始めると回しやすい。

- 主要 SaaS を台帳にする
- 承認済み、要審査、禁止を分ける
- 重要 SaaS の OAuth 連携を棚卸しする
- 外部共有設定を一つずつ確認する
- 解約前チェック欄を作る

台帳に最初から全部を入れなくてもよい。まずは、顧客データ、人事データ、会計データ、社内ファイルに触るものから始める。そこに管理者、認証方式、主要連携、export 可否の欄を足すだけでも、かなり実務が軽くなる。

また、例外を減らすことも大切である。部門ごとに別ツール、担当者ごとに別連携、管理者ごとに別ルールが増えると、SaaS　の数以上に運用が膨らむ。小さな会社ほど、使うサービスと接続方式を絞った方がよい。

## 通常時の例と、崩れた時の例

通常時の例では、ひとり情シスが主要　SaaS　台帳を持っている。そこには、用途、管理者、認証方式、取り扱いデータ、主要連携、外部共有、更新日、出口欄が入っている。現場が新しい SaaS を使いたいと言ったら、まず承認済みサービ

スで代替できないかを見る。必要なら要審査として、データ種別、OAuth 連携、export 可否、共有境界を確認する。Google Workspace や Microsoft 365 の連携アプリは定期的に見直し、不要な同意は外す。結果として、使うサービスは増えても、何がどこへ届くかを説明できる。

崩れた時の例では、契約一覧はあるが、利用実態が分からない。現場は勝手に SaaS を試し、Google や Microsoft アカウントでログインして、便利そうな連携にそのまま同意する。共有設定は既定値のままで、誰でも外部共有できる。いざ退職者対応や解約の話になると、どのアプリを止めるべきか、どこにデータがあるか、何が export できるかが分からない。結果として、SaaS が増えるほど、会社側の把握が追いつかなくなる。

## 最低限ここまではやる

第12章の内容をすべて一度に整えなくてもよい。だが、次の四つは早めに持ちたい。

1. 主要 SaaS の運用台帳を持つ
2. 承認済み、要審査、禁止の三分けを作る
3. OAuth や API 連携を別管理する
4. 解約前にデータ出口を確認する

この四つがあるだけで、SaaS 管理はかなり崩れにくくなる。SaaS 管理とは、何を契約したかを覚えることではない。誰が使い、何と連携し、どこへ共有し、どう終わらせるかを会社として把握し続けることである。

## 今日、今週、後でやること

今日やることは、主要 SaaS を五つだけ選び、用途、管理者、認証方式、主要連携を書き出すことである。空欄が多いなら、そこが最初の詰まりどころである。

今週やることは、その中で一つの Google Workspace 連携か Microsoft 365 連携を見て、どのアプリが何の権限を持っているか確認することである。ここを見るだけでも、連携アプリ管理の感覚が変わる。

後でやることは、主要 SaaS の外部共有設定と export 可否を順に確認することである。入口と出口が見えると、SaaS は初めて会社の管理対象になる。

第13章

# メール、チャット、ファイル共有の運用

会社の問い合わせ窓口は一応ある。`sales@` も `support@` も存在する。だが実態は、どちらも特定の担当者の個人メールへ転送されている。日中の相談はチャットで流れ、取引先へ送る最新版資料は添付で往復し、誰が何を確定したのかは後から追いにくい。担当者が休むと返事が止まり、退職すると過去経緯の所在が分からなくなる。

メール、チャット、ファイル共有は、いまやどの会社にもある。だからこそ整備されているように見えやすい。だが実際には、最も日常的に使う基盤ほど、慣習だけで回りやすく、個人依存しやすい。第13章で扱うのは、ここをどう崩れにくくするかである。

第12章では SaaS やクラウドの統制を扱った。第13章では、その上で毎日の連絡と共有をどう運用するかを見る。論点は単純である。どの手段を何に使うか。共有窓口をどう持つか。社外とどこまでつなぐか。誤送信をどう減らすか。この四つが曖昧だと、どれほど高価なツールを入れても運用品質は上がらない。

## メール、チャット、ファイル共有の役割を分ける

最初に決めたいのは、どの道具を何に使うかである。ここが曖昧だと、連絡は速くなっても、後から探せない、代理対応できない、最新版が分からない、という問題が残る。

大きく分けると、役割は次のように整理しやすい。

- メールは、対外連絡、正式依頼、承認依頼、後で追いたいやりとりに使う

- チャットは、短い相談、状況確認、即時連携、軽い調整に使う
- ファイル共有は、最新版の保管、共同編集、社内外への閲覧提供に使う

これは絶対的なルールではない。だが、基準がないとすべてがチャットへ流れ込み、重要な指示も決定も埋もれやすい。逆に、何でもメールに寄せると、最新版の資料まで添付で往復し、誰がどの版を見ているか分からなくなる。

ひとり情シスとしては、各手段の使い分けを厳密な規程にする必要はない。まずは、次の三点だけでも共有しておくとよい。

1. 社外への正式連絡はメールを基本とする
2. 最新版を残すものは添付でなく共有先を基準にする
3. チャットで決まった重要事項は、後から見つけられる場所へ戻す

この三つがあるだけで、連絡基盤の散らかり方はかなり変わる。

## 代表アドレスと共有対応は個人メールで回さない

第13章で最も強く言いたいのはここである。問い合わせ、採用、請求、総務連絡のような会社窓口を、個人メールへの転送で回してはいけない。

理由は単純である。個人メールに寄せると、その人の在席が運用の前提になるからだ。休暇、退職、異動、委託先交代のたびに、転送設定、代理送信、過去経緯の確認で詰まる。第9章で退職時のメール処理を扱ったが、本来はその前に、会社窓口を個人から切り離しておくべきである。

実務上は、環境に応じて次のような型を取る。

- Microsoft 365 なら shared mailbox を使う
- Google Workspace なら delegation や collaborative inbox を使う

- どうしても個人受信箱を使う期間があるなら、暫定措置だと明示し、早めに共有運用へ移す

ここで重要なのは、パスワード共有をしないことである。複数人で見る必要があるからといって、一つのメールアカウントを共用してはいけない。監査証跡が消え、退職時や委託先変更時に制御できなくなる。

共有窓口では、誰が閲覧できるかだけでなく、どの名義で返信するかも決めておきたい。たとえば、`support@` として送るのか、`山田 via support@`のように代理送信だと分かる形にするのかで、受け手の印象も、内部の追跡性も変わる。小さな会社では細かな設計まで難しく見えるが、少なくとも次は決めた方がよい。

- どのアドレスを共有窓口にするか
- 誰が見られるか
- 誰が送れるか
- 誰が管理者か
- 不在時は誰が一次対応するか

問い合わせ窓口が個人依存から外れるだけで、会社の連絡基盤はかなり強くなる。

## チャットの外部協業は、接続モデルを決めてから開く

社外とチャットできる機能は便利である。だが便利だからといって、同じ感覚で外部接続を増やすと、どこまで見えているのかが曖昧になる。

たとえば Microsoft Teams では、外部とつながる方法が一つではない。external access は主にチャットや通話の相互接続であり、相手を自社の Teams や SharePoint の内部空間へ入れるものではない。一方で guest access は、相手をチームやチャネル、ファイル空間へ参加させる協業モデルである。同じ「外部と Teams でやりとりする」でも、見せる範囲は大きく違う。

Google Chat や Slack でも同じである。外部ユーザーと一対一で話すだけなのか、スペースやチャネルを共有するのかで、運用上の重さは変わる。案件が終わった後に外部参加者を残したままにしないためにも、最初に接続モデルを分けた方がよい。

実務では、社外チャットを次の三つに分けると整理しやすい。

- 単発連絡
- 案件期間中の協業
- 恒常的な委託先、保守会社、顧問との連携

単発連絡なら、外部チャットよりメールへ戻した方が管理しやすいことも多い。案件期間中の協業なら、外部参加のあるチャネルやスペースを明示し、終了日を意識する。恒常的な連携なら、接続先を一覧化し、誰が承認し、誰が見直すかを決める。

特に注意したいのは、設定を閉じても過去の外部参加者や共有ファイルが自動で片付くわけではない点である。Google Chat でも、外部チャットの無効化と既存の外部ユーザー除去は別作業である。つまり、社外とのチャット運用は「開ける設定」だけでなく、「終わらせる手順」まで含めて設計する必要がある。

## ファイル共有は、リンクより境界と期限を決める

ファイル共有でもっとも避けたいのは、最新版が添付とリンクの両方に散らばることである。見積書、提案書、手順書のように更新されるものは、基本的には共有先を正本にした方がよい。

ただし、共有リンクなら何でもよいわけではない。第13章で重視したいのは、共有のしやすさより、誰がどこまでいつまで見られるかである。最低限、次の四点は意識したい。

- 対象者は誰か
- 相手の認証は必要か
- 期限を付けるか
- 再共有を許すか

重要な資料ほど、認証のある共有を基本にした方がよい。誰かがリンクを転送しただけで読める状態は、便利ではあるが追跡しにくい。期限を付けられるなら付ける。編集者が自由に再共有できる設定も、必要な場面だけに絞る。これだけでも、社外共有の事故は減る。

もう一つ見落としやすいのが、チャットとファイル共有のつながりである。たとえば Google Drive では、Chat スペースに共有したファイルが、後からそのスペースへ参加した人にも見える場合がある。つまり、「このファイルを渡した」のではなく、「この場に入る人へ見える状態を作った」ことになる。Teams や Slack でも似たことが起きる。ファイルを置く場所と会話の場所が結びついているからだ。

だから、社外と共有するファイルは、案件用、委託先用、社内限定用のように領域を分けた方がよい。誰にでも同じ共有ドライブや同じチームサイトを使わせると、境界があいまいになる。

## 誤送信と誤共有は、送る前に防ぐ

誤送信対策というと、「送ってしまった後に取り消せるか」を考えたくなる。だが実務では、そこに期待しない方がよい。たとえば Microsoft の cloud-based message recall は、同一組織内の Exchange Online 環境での条件が大きく、社外宛てには頼れない。つまり、誤送信対策の主戦場は送信前である。

送信前に効く対策は、派手ではないが実用的である。

- 社外宛先への警告を出す
- 大量宛先や配布先への注意を出す
- 宛先を最後に見直す
- 添付が本当に必要かを確認する
- 共有リンクの公開範囲を送信前に確認する

ここで大切なのは、確認項目を増やしすぎないことである。五項目も十項目もあると誰も見なくなる。小さな会社なら、次の三つで十分である。

1. 外部宛先は含まれていないか
2. 添付やリンクの中身は合っているか
3. この相手に本当にこの範囲まで見せてよいか

MailTips のような事前警告が使える環境なら活用した方がよい。だがそれだけで防げるわけではない。結局は、送る前に一呼吸置く運用を組み込めるかどうかである。

## 代理対応、引き継ぎ、不在時運用に崩れない設計

連絡基盤の品質は、平常時より、崩れた時に分かる。担当者が休む。退職する。委託先が変わる。災害や障害で普段の連絡先へ届かない。そういう時に止まる設計は、最初から脆い。

崩れにくくするためには、日常運用の時点で次を共有化しておきたい。

- 顧客や取引先とつながる代表アドレス
- 案件の主要チャネル
- 最新版を置くファイル置き場
- 重要な決定事項を残す場所

ここで言う共有化は、全員が何でも見られるようにすることではない。担当を明確にしつつ、個人一人がいなくなっても止まらない状態にすることである。たとえば、営業問い合わせは shared mailbox へ入る。一次返信担当は営業リーダーと情シス補助の二名にする。見積テンプレートや最新資料は共有フォルダへ置く。チャットで決まったことは案件メモへ戻す。こうしておけば、誰か一人が不在でも業務は続く。

第9章では退職時の停止、失効、引き継ぎを扱った。第13章で伝えたいのは、その作業を軽くする鍵は、通常時から共有窓口へ寄せることだという点である。退職者メールボックスから過去経緯を掘り出す運用は、あくまで救済策であって標準ではない。

## 小さな会社で現実的に回るやり方

すべての会社が、高度なメール保護や外部協業管理製品を入れられるわけではない。小さな会社では、まず例外を減らす方が効く。

現実的には、次のくらいから始めるとよい。

- 代表アドレスを三つだけ共有運用にする
- 取引先との恒常窓口は個人メールへ転送しない
- 社外チャットは案件単位で許可し、終了時に見直す
- 共有ファイルは、社外向け領域を分け、期限付きリンクを基本にする
- 重要事項はチャットだけで閉じず、後から見つかる場所へ残す

この章のポイントは、完璧な統制ではない。迷った時に戻る基準を持つことである。たとえば、判断に迷うなら次のように考えるとよい。

- 社外との正式な依頼や合意ならメールへ戻す
- 最新版を参照してほしいなら添付でなく共有先を渡す
- 外部参加者を中へ入れる必要があるか疑わしいなら、まずは外部チャットやメールで始める

単純な基準がある方が、現場は回る。

## 通常時の例と、崩れた時の例

通常時の例では、問い合わせ窓口、採用窓口、請求窓口が共有メールボックスまたは共有受信箱で管理されている。担当者は複数名おり、送信名義も決まっている。社外との案件チャネルは、案件単位で作られ、終了時に外部参加者を見直

す。最新版資料は共有フォルダか共有ドライブに置き、メールではリンクを送る。社外共有は認証必須を基本にし、必要に応じて期限を付ける。誤送信防止は、外部宛先、添付、共有範囲の三点確認で回している。結果として、担当者が休んでも、顧客対応と案件進行は止まらない。

崩れた時の例では、`support@` は担当者一人の個人メールへ転送されている。顧客とのやりとりは個人受信箱に散らばり、最新資料は添付で往復している。社外とのやりとりは個別のチャットで進み、誰が外部参加者として残っているか分からない。退職や引き継ぎのたびに、メール転送、共有リンク、過去ファイルの所在確認が必要になる。結果として、道具はそろっているのに、業務は人について回る。

## 最低限ここまではやる

第13章の内容を一度にすべて整えなくてもよい。だが、次の四つは早めに持ちたい。

1. 代表アドレスを個人メール依存から外す
2. メール、チャット、ファイル共有の役割を決める
3. 社外チャットの接続モデルを分ける
4. 誤送信前の三点確認を決める

この四つがあるだけで、日常運用の事故と引き継ぎ負荷はかなり減る。メール、チャット、ファイル共有の運用とは、便利に連絡できることではない。誰が不在でも、どこまで社外とつながっても、業務が止まらず、後から追える状態を保つことである。

## 今日、今週、後でやること

今日やることは、会社窓口として使っているメールアドレスを洗い出し、そのうち個人メールへ転送されているものを確認することである。一つでも見つかれば、そこが最初の改善点である。

今週やることは、代表アドレスを一つだけ共有運用へ移すことである。shared mailbox、delegation、collaborative inbox のどれでもよい。とにかく、個人依存を一つ減らす。

後でやることは、社外チャットと社外共有リンクの運用を見直し、案件終了時の見直し手順と、共有期限の標準を決めることである。ここまでできると、連絡基盤は単なる便利ツールから業務基盤へ変わる。

第14章

# サーバー、Web、ドメイン、DNS、証明書

会社の Web サイトは普通に見えている。採用ページも、お問い合わせフォームも、一応動いている。だから問題はないように見える。ところが、ドメイン更新日を聞くと誰も答えられない。DNS は制作会社が管理しているらしいが、どこのサービスか分からない。証明書更新は「たぶん自動です」と言われるが、何がどう自動なのかを社内で説明できる人がいない。こういう状態は珍しくない。

公開資産は、普段静かである。社内端末のように毎日触るわけでもなく、ネットワークのように不調が目立つわけでもない。だからこそ、責任者不明のまま放置されやすい。だが、ドメインが切れればサイトもメールも止まる。DNS を誤れば閲覧先が変わる。証明書が切れれば利用者は不安になる。公開サーバーの更新が止まれば侵入口になる。第14章で扱うのは、この静かな資産をどう見失わないかである。

第11章では接続基盤、第12章ではクラウド統制、第13章では日常の連絡基盤を扱った。第14章では、その外側にある公開基盤を見る。論点は、サーバーを持つかどうかより先に、公開資産の棚卸し、責任分界、期限管理、変更手順である。

## 公開資産の管理は、サーバーの前に一覧化から始める

第14章で最初に持ちたいのは、公開資産台帳である。サーバーがある会社だけでなく、外注サイトや SaaS 型 CMS を使っている会社でも必要だ。

最低限、次は見えるようにしたい。

- ドメイン名
- 主なサブドメイン
- レジストラ
- DNS ホスト
- ホスティング先
- Web アプリや CMS
- 証明書の更新方式
- 管理者
- 契約更新日や有効期限

ここで重要なのは、会社サイト という一項目で終わらせないことである。実際には、本体サイト以外にも次のようなものがぶら下がっている。

- 採用サイト
- キャンペーン用 LP
- 外注フォーム
- 旧製品サイト
- 委託先向けポータル
- 検証用のまま残ったサブドメイン

公開資産の事故は、主力サイトより、こうした周辺から起きやすい。理由は単純で、誰も見ていないからだ。だから一覧化では、重要資産だけでなく、古いもの、止めたつもりのもの、担当不明のものを拾う必要がある。

## オンプレとクラウドの使い分けは、維持管理責任で決める

公開基盤を考える時、つい「オンプレかクラウドか」という言い方をしたくなる。だが、ひとり情シスにとって本当に重要なのは、誰が更新責任を持つかである。

NIST　のサーバー保護資料が示す通り、サーバーを持つということは、選定、実装、更新、監視を持つということである。OS、Web　サーバー、ランタイム、ミドルウェア、CMS、バックアップ、ログ、証明書。これらを継続して見られないなら、自前の公開サーバーは重い。

そのため、小さな会社では次の順で考える方が現実的である。

- static hosting や managed hosting で足りないか
- SaaS 型 CMS やノーコード公開基盤で足りないか
- それでも足りないなら VPS や自前サーバーが必要か

自前サーバーが必要になるのは、独自アプリ、特殊なミドルウェア、細かな制御、社内要件など、持つ理由が明確な時である。逆に、会社案内、採用ページ、問い合わせ窓口程度なら、まずは運用負荷の少ない形を優先した方がよい。

ここでの結論は単純だ。公開サーバーは、作れるかではなく、維持できるかで決める。持つ理由が弱いのにサーバーだけ増えると、後で更新漏れの温床になる。

## ドメインと DNS は、契約先と責任分界を分けて管理する

公開資産で特に混乱しやすいのが、ドメインと DNS を同じものだと思ってしまうことである。実際には、次は別かもしれない。

- ドメインを登録しているレジストラ
- DNS レコードを持っている DNS ホスト
- Web サイトやアプリを置いているホスティング先

この三つが別でも普通である。だからこそ、どこにログインすれば何を変えられるか を分けて把握しておきたい。

ドメイン管理でまず押さえたいのは、更新漏れを防ぐことだ。ICANN は、レジストラが有効期限の約1か月前と約1週間前に更新通知を送ること、auto-renew の活用、連絡先を最新に保つことを重視している。つまり、ドメイン運用の基本は次の三つである。

1. auto-renew を有効にする
2. 支払い情報を最新にする
3. 更新通知を個人メールでなく共有管理する

ここで注意したいのは、auto-renew があるから安心とは限らない点である。カード失効、登録メール未着、担当者退職で止まることは普通にある。だから、期限の見える化と通知先の共有化は別途必要である。

DNS 側では、少なくとも主要レコードの意味を説明できる状態にしておきたい。

- A、AAAA は Web やアプリの接続先 IP を持つ
- CNAME は別名参照に使う

- MX はメール配送先を示す
- TXT は各種検証や認証に使われる
- NS はそのドメインをどのネームサーバーが持つかを示す

細かな設計を全部覚える必要はない。だが、どのレコードがどの用途か分からないまま触ると、Web だけでなくメールや認証まで巻き込む。第14章では、DNS を専門家の黒箱にしないことが重要である。

## DNS 切替は、事前準備と待ち時間がある変更作業である

DNS 変更でありがちなのは、「レコードを書き換えたのにすぐ変わらない」「戻したのに一部だけおかしい」という混乱である。これは DNS の仕組みというより、事前準備不足で起きることが多い。

Cloudflare の資料でも、TTL は DNS resolver がどれだけ長くキャッシュするかを制御する項目だと整理されている。AWS Route 53 の移行手順では、現在設定の取得、TTL の事前調整、古い TTL の期限切れ待ち、NS 切替、監視、必要なら切り戻し、という順番が明示されている。つまり、DNS 切替はその場の設定変更ではなく、変更管理そのものである。

実務では、DNS 変更前に最低限次を持ちたい。

- 現在の DNS 設定控え
- ゾーンファイルやレコード一覧
- 変更対象レコード
- 変更日時
- TTL の事前調整計画
- 切り戻し先
- 変更後の確認項目

ここで特に重要なのは、TTL を事前に下げるという発想である。切替直前に TTL を短くしても、すでに古い TTL でキャッシュされている利用者にはすぐ効かない。だから、早めに準備し、古い TTL が抜ける時間を見込む必要がある。

また、DNSSEC を使っている場合は、DS レコードの扱いも別論点になる。すべての会社で必要ではないが、使っているなら「DNS を移すだけ」で済まない。高度な設定ほど、移行手順を軽く見ない方がよい。

## Web サイトと公開アプリは、公開後の運用責任が本体である

Web サイトや公開アプリは、作る時にだけ注目されやすい。だが、実際に危ういのは公開後である。古い特設サイト、更新されないフォーム、止まった採用ページ、誰も入らない管理画面が残ると、公開資産は一気に危うくなる。

ここで持ちたい視点は、公開物を案件でなく運用で見ることだ。たとえば WordPress のような公開 CMS を置くなら、見るべきものは次である。

- WordPress core の更新
- plugin の更新
- theme の更新
- 更新前バックアップ
- 不要 plugin の整理
- trusted sources からの導入

WordPress 公式資料は、常に最新バージョンへ更新すること、古い版はセキュリティ更新対象外であること、plugin 更新前には current backup を持つべきことを明確に示している。つまり、公開 CMS は作成時より、更新運用の方が本体である。

ここで外注と内製の別はあまり関係ない。外注先が作ったサイトでも、公開後の責任がどこにあるかを社内で説明できなければ危うい。ベンダー保守契約があるなら、その範囲、緊急連絡先、更新手順を持つ。ないなら、自社で更新できる体制か、そもそもその置き方が妥当かを見直す必要がある。

## 証明書更新と期限管理は、自動化を前提にする

証明書運用で一番避けたいのは、担当者の記憶に依存することである。証明書は切れた瞬間に利用者が不安になり、社内では障害扱いになる。だから、手動更新を平常運転にしない方がよい。

Let's Encrypt は 90 日証明書を前提にし、自動更新を基本思想としている。これは厳しすぎる運用ではなく、むしろ「人が覚えていなくても回る」ための設計である。第14章では、証明書運用で最低限次を記録したい。

- どの証明書を使っているか
- どの方式で発行しているか
- 自動更新か手動更新か
- 更新失敗時に誰へ通知するか
- 関連する DNS やサーバー設定は何か

もし 年に一回、担当者が思い出したら更新する という運用が残っているなら、それは技術的負債である。すぐに置き換えられなくても、少なくとも期限監視と引き継ぎ可能な手順は必要だ。

証明書では、更新できることだけでなく、更新失敗に気づけることも重要である。自動更新ジョブがあっても、失敗通知が誰にも届かないなら意味が薄い。自動化と監視はセットで考えたい。

## 公開系システムで注意すべき運用

公開系では、技術そのものより、管理の置き方で事故が起きやすい。特に注意したいのは次の点である。

- 個人アカウント依存を避ける
- 外注任せにしすぎない
- 変更記録を残す
- 不要サブドメインを残さない
- ドメイン移管を期限直前にやらない

個人依存の典型は、昔の担当者の個人メールがレジストラ連絡先になっている状態である。これは第9章、第13章と同じ問題で、公開基盤でも強く起きる。更新通知、請求、証明書失敗通知が個人へ飛ぶ設計は避けたい。

また、外注任せにしすぎるのも危うい。制作会社や保守会社が実務を担うこと自体はよい。だが、どのドメインがどこにあり、誰が最終責任者か　を社内で説明できない状態は避けるべきである。

変更記録も大切である。DNS をいつ変えたか、証明書をどの方式へ切り替えたか、Web の公開先をどこへ移したか。これを残しておかないと、次の変更で毎回調べ直すことになる。

## 小さな会社で現実的に回るやり方

すべての公開資産に高度な監視や二重化を入れられる会社ばかりではない。小さな会社では、まず見える化と例外削減の方が効く。

現実的には、次のくらいから始めるとよい。

- 公開資産台帳を作る
- 主要ドメインの auto-renew を確認する
- 更新通知先を共有メールへ寄せる
- 主要サイトの証明書期限を月1回確認する
- WordPress サイトの更新日を残す
- DNS 変更前に控えと切り戻し先を持つ

ここで意図的に避けたいのは、公開資産の種類を増やしすぎることである。古い LP、検証用サブドメイン、用途の薄い独立サイトが増えるほど、期限管理と更新責任が広がる。小さな会社ほど、公開先を絞る方が運用しやすい。

## 通常時の例と、崩れた時の例

通常時の例では、会社は公開資産台帳を持っている。そこには、本体ドメイン、採用サブドメイン、問い合わせフォーム、DNS　ホスト、レジストラ、証明書方式、管理者、更新通知先が入っている。主要ドメインは　auto-renew　で、通知は共有アドレスに届く。DNS　変更時には事前にレコード控えを取り、TTL　を調整し、変更後確認と切り戻し先を準備する。WordPress サイトは core と plugin の更新日が残っており、更新前バックアップも確認している。結果として、公開基盤の変更や引き継ぎが人の記憶に依存しない。

崩れた時の例では、ドメインは昔の制作担当者が取得したままで、レジストラも　DNS　ホストも分からない。証明書は「たぶん自動」と言われるが、方式も通知先も不明である。古いキャンペーンサイトや使っていないサブドメインが残り、WordPress plugin 更新も止まっている。DNS 切替はその場で行われ、元設定の控えもなく、何か起きると戻せない。結果として、公開資産は見えているのに管理されていない。

## 最低限ここまではやる

第14章の内容をすべて一度に整えなくてもよい。だが、次の四つは早めに持ちたい。

1. 公開資産台帳を作る
2. 主要ドメインの更新通知を共有管理する
3. DNS 変更前の控えと切り戻し先を持つ
4. 証明書更新方式と期限監視を確認する

この四つがあるだけで、公開資産の事故はかなり減る。公開資産の管理とは、Web サイトが見えることではない。どのドメインが、どの DNS で、どの証明書で、誰の責任で動いているかを会社として把握し続けることである。

## 今日、今週、後でやること

今日やることは、会社が使っている公開ドメインとサブドメインを洗い出すことである。本体サイトだけでなく、採用、問い合わせ、古い LP、検証用まで含める。

今週やることは、その中で重要なドメインを三つ選び、レジストラ、DNS ホスト、証明書更新方式、通知先を確認することである。空欄が多いなら、そこが公開基盤の弱点である。

後でやることは、WordPress を含む公開サイトの更新体制と、DNS 変更時の標準手順を一枚にまとめることである。ここまでできると、公開資産は「昔から動いているもの」ではなく、会社が管理している資産へ変わる。

第15章

# データ保護、バックアップ、ログ、保持

バックアップは取ってある。そう聞くと少し安心する。ところが実際に「昨日の状態へ戻せますか」「この共有フォルダだけ復元できますか」「半年前の管理者操作ログを追えますか」と聞くと、急に答えがあいまいになることがある。NAS にコピーはある。SaaS に retention はある。監査ログもあるはずだ。だが、戻したことはない。保持期間は知らない。検索方法も決まっていない。これは珍しい状態ではない。

データ保護が難しいのは、似た言葉が多いからである。バックアップ、保持、ログ、アーカイブ、削除保留。どれも「残す」話に見える。だが、目的は同じではない。バックアップは復旧のためにある。保持は保存と削除のルールのためにある。ログは追跡と調査のためにある。この違いを曖昧にしたまま運用すると、平時は静かでも、有事に弱い。

第15章で扱うのは、ここを分けて考えるための型である。何を守るのか。どこまで戻せればよいのか。何のログをどれだけ残すのか。保存と削除をどう設計するのか。そして、本当に戻せることをどう確認するのか。この順番で見ていく。

## データ保護は、何を守るかを分けるところから始まる

最初に決めたいのは、何を守るかである。ストレージを丸ごと守る発想から入ると、重要度の違うものが全部同じ扱いになり、運用が重くなる。

まずは、次のように分けて考えると整理しやすい。

- 売上、受発注、顧客対応に直結する業務データ

- 人事、会計、契約などの機微データ
- 共有ファイルやナレッジ
- システム設定、スクリプト、構成情報
- 監査ログ、操作ログ、障害ログ

CISA も、バックアップは何を守るかの inventory から始めるべきだとしている。つまり、バックアップの出発点は保存先ではなく、業務影響である。

ここで強く意識したいのは、復旧優先順位である。全部が同じ重さではない。たとえば、受発注データは当日中に戻したいが、過去の広報画像は翌週でもよいかもしれない。管理者操作ログは量が多くても、事故調査では重要かもしれない。だから、「何を残すか」と「どの順で戻すか」を一緒に決めた方がよい。

難しい言葉を使えば、どこまで巻き戻せればよいか と どれくらいで戻したいか である。だが、ひとり情シスの現場では、まず日本語で十分だ。昨日まで戻せればよいのか。数時間前まで必要か。半日止まってもよいのか。1 時間以内に戻したいのか。この粒度で考えるだけでも、設計はかなり変わる。

## バックアップは、復旧目標から設計する

バックアップを「毎日取っています」で終わらせると危うい。重要なのは、その頻度と方式で本当に必要な復旧ができるかどうかである。

たとえば、日次バックアップだけでは、午前中の入力を午後に失うかもしれない。逆に、毎時間バックアップを取っていても、戻すのに丸一日かかるなら、実務では困るかもしれない。だから先に見たいのは、次の点である。

- どこまで前の状態へ戻せればよいか
- どれくらいの時間で戻したいか
- ファイル単位で戻したいのか

- システム単位で戻したいのか

NIST の contingency planning は、復旧を計画、手順、技術的措置を含む coordinated strategy として扱っている。これは実務的である。バックアップも同じで、媒体やソフトだけでなく、優先順位、手順、代替手段まで含めて初めて意味を持つ。

ここでよくある誤解は、「全部守ろう」とすることである。実際には、重要度に応じて差をつけた方が回る。たとえば次のような分け方がある。

- 基幹データは高頻度で取り、優先復旧対象にする
- 共有ファイルは日次と世代管理を基本にする
- 構成情報やスクリプトは更新のたびに残す
- 一時ファイルや再生成可能なデータは軽く扱う

重要なのは、守る範囲を絞ることではなく、重さを分けることである。

## 世代管理、別媒体、別ネットワーク保管を組み合わせる

バックアップで一番避けたいのは、同じ事故で本体とバックアップが一緒にやられることである。ランサムウェア、誤削除、設定ミス、物理故障。原因は違っても、一か所依存は弱い。

CISA は、offline かつ encrypted backups を維持し、regularly test することを勧めている。さらに golden images の維持も挙げている。つまり、データ保護では次の組み合わせが重要になる。

- 複数世代を持つ
- 本体と別媒体に置く
- 本体と別ネットワークに置く

- 必要に応じてオフライン化する
- 復旧用の土台も残す

ここで言う世代管理とは、最新一個だけを持つことではない。最新版が壊れても、一つ前、二つ前へ戻れるようにする考え方である。誤削除や改ざんは、気づいた時にはすでに最新バックアップにも反映されていることがあるからだ。

また、分離も重要である。同じ認証情報、同じネットワーク、同じ管理画面で本体とバックアップがつながっていると、侵害時にまとめて届かれる。小さな会社でも、少なくとも「バックアップ先は普段の利用者から直接見えない」「削除や暗号化にすぐ届かない」状態は作りたい。

golden image の考え方も有効である。これは、サーバーや重要端末を再構築するための土台を別に持つ発想だ。データだけ戻せても、OS や基本ソフトを戻せないと復旧は遅くなる。第15章では、バックアップ対象をデータだけに限らず、再構築の土台まで含めて見たい。

## ファイルサーバーと SaaS は同じ守り方をしない

ここはひとり情シスが迷いやすい点である。ファイルサーバーの感覚で SaaS を見たり、逆に SaaS の retention をそのままバックアップだと思ってしまったりする。

ファイルサーバーや NAS では、一般に次の発想で守りやすい。

- フォルダ単位やボリューム単位でコピーを持つ
- 世代を持つ
- 必要ならファイル単位で戻す

一方で SaaS は、サービスごとに守り方が違う。Microsoft 365 や Google Workspace には retention、recycle bin、audit log、export の機能があるが、それぞれ目的が違う。Microsoft Purview の retention は retain-only、delete-only、retain and then delete のような保持ルールであり、content in place で動く。Google Vault も、削除後も searchable and exportable に保つことができる。だが、これは復旧目的の別コピーと同一ではない。

ここでの実務的な整理は次である。

- ファイルサーバーはバックアップ方式を明示する
- SaaS は native retention、recycle bin、export、監査ログの範囲を確認する
- その上で、足りない復旧要件があるなら別対策を考える

この「足りない復旧要件」という考え方が重要である。すべての SaaS に追加バックアップ製品が必要とは限らない。だが、監査ログが 6 か月しか残らない、削除後の戻し方が限定的、ファイル単位でなくサービス単位でしか戻せない、という差は普通にある。だから、ファイルサーバーと SaaS を同じ言葉で片付けない方がよい。

## ログ取得、保持、検索の設計

ログは、残っているだけでは足りない。NIST SP 800-92 が示すように、ログ管理は generate、transmit、store、analyze、dispose まで含む。つまり、次の四つを決めないと、有事に使いにくい。

- 何を取るか
- どこへ集めるか
- どれだけ残すか
- 誰が見られるか

ここでログを大きく分けると、少なくとも次がある。

- 監査ログ
- 管理者操作ログ
- 障害ログ
- セキュリティ関連ログ

全部を同じ保存先、同じ保持年数、同じ閲覧権限で持つ必要はない。むしろ分けた方がよい。たとえば、障害切り分け用のアプリログは短めでもよいが、管理者操作や認証の監査ログは長く持ちたいかもしれない。

ベンダー標準の保持期間も見落としやすい。Microsoft Purview では、Audit (Standard) は 180 日、Audit (Premium) では Exchange、SharePoint、OneDrive、Microsoft Entra の監査記録が 1 年保持される。一方で Google Workspace では、明記のない多くの log events が generally 6 months、Email log search は 30 days、Vault log events は indefinite である。つまり、「必要になったら後で見ればよい」は危うい。

ログ設計で特に大切なのは、検索導線である。検索画面がどこにあるか、どの権限で見られるか、CSV に出せるか、どこまで遡れるか。これが決まっていないと、ログが存在しても調査は遅れる。

## 保持は、保存と削除をセットで考える

保持設定は安全そうに見えるが、同時に削除設定でもある。ここを軽く見ると危うい。

Microsoft Learn は、retain-only、delete-only、retain and then delete の三つを明確に分けている。さらに retention wins over deletion の原則を示している。つまり、保持設定は、単に残すためだけでなく、いつ消すか、何が優先するかを決める仕組みである。

Google Vault も同じで、indefinite retention にすれば削除後も searchable and exportable な状態を保てる一方で、retention period の設定次第では、ルール適用時に古いデータが immediately purge されうると警告している。これはとても重要である。保持ルールは、後から静かに効くのではなく、設定した瞬間に影響することがある。

だから保持設計では、次を分けて考えたい。

- とにかく一定期間は消してはいけないもの
- 古くなったら消したいもの
- 調査や法務の理由で例外的に残すもの

ここで「全部永久保存」が安全に見えることもある。だが、実務では逆に使いにくくなる。ノイズが増え、検索性が落ち、削除すべきものまで残る。第15章では、保存と削除を両方設計する方がよいと伝えたい。

## 復旧を前提にした運用確認

復元テストをしていないバックアップは、まだ完成していない。これは第15章の中心メッセージの一つである。

CISA も、バックアップは regular basis で availability と integrity を test すべきとしている。理由は単純だ。取れていると思っていたが失敗していた、復元に必要な手順が欠けていた、権限が足りず戻せなかった、ということが普通に起きるからである。

復元テストは大掛かりでなくてもよい。小さな会社なら、まずは次のくらいから始められる。

- 共有フォルダのサンプルファイルを 1 つ戻す
- 主要 SaaS の検索や export を 1 回試す
- 管理者操作ログを 1 件検索してみる
- 復旧手順の空欄を埋める

ここで大切なのは、本番障害が起きてから初めて触らないことである。戻す練習を一度でもしておくと、必要な権限、時間、詰まりどころが見える。逆に、練習していないと、有事に「残っているか」だけでなく「どう戻すか」から調べることになる。

また、復旧確認は年一回の儀式にしない方がよい。すべてを毎月試す必要はないが、対象を回しながら小さく継続した方が運用しやすい。

## 小さな会社で現実的に回るやり方

すべてのログを長期保管し、すべてのデータを多重化し、頻繁に大規模復旧訓練をするのは現実的ではない会社も多い。小さな会社では、まず重要なものに絞る方が効く。

現実的には、次のくらいから始めるとよい。

- 重要データ一覧を作る
- バックアップ対象、頻度、世代、保管先を書く
- オフラインまたは分離保管を一つ持つ
- 主要 SaaS のログ保持期間を確認する
- 保持ルールの一覧を持つ

- 月に一つだけ復元や検索を試す

ここで意図的に避けたいのは、何でも残すことと、何も試さないことである。前者は管理不能になり、後者は有事に役立たない。小さな会社ほど、「重要なものを、必要な期間だけ、戻せる形で持つ」という単純な方針の方が強い。

## 通常時の例と、崩れた時の例

通常時の例では、会社が重要データ一覧を持っている。受発注データ、人事データ、共有ファイル、主要　SaaS、管理者監査ログが載っており、それぞれに復旧優先順位、バックアップ方式、保持期間、担当者が書かれている。ファイルサーバーは世代付きで分離保管され、主要 SaaS は native retention と export 可否が確認済みで、監査ログ保持期間も把握している。月に一度は何か一つ復元や検索を試し、手順を更新している。結果として、「何をどこまで守れているか」を説明できる。

崩れた時の例では、バックアップはあると言われるが、対象も世代も保管先も説明できない。SaaS は retention があるから大丈夫だと思っているが、どのデータがどれくらい残るかは知らない。監査ログも「たぶん残る」と考えていたが、必要になった時には保持期間を過ぎている。保持ルールを変えたら古いメールが消え、復元手順も分からない。問題は機能不足ではなく、目的を分けずに運用していたことである。

## 最低限ここまではやる

第15章の内容を一度にすべて整えなくてもよい。だが、次の四つは早めに持ちたい。

1. 重要データと復旧優先順位を決める

2. バックアップ対象、頻度、世代、保管先を書く
3. 主要ログの保持期間を把握する
4. 小さくても復元テストを実施する

この四つがあるだけで、データ保護はかなり実務的になる。データ保護とは、たくさん保存することではない。何をどれだけ残し、どこまで追え、どの順で戻せるかを会社として決めておくことである。

## 今日、今週、後でやること

今日やることは、守るべきデータを五つだけ書き出し、それぞれに「止まると何が困るか」を一言添えることである。これだけで、優先順位の感覚が出てくる。

今週やることは、その中から一つを選び、実際のバックアップ対象、保持期間、保管先、復元方法を書き出すことである。同時に、主要 SaaS の監査ログ保持期間も一つ確認したい。

後でやることは、月次の小さな復元テストと、保持ルール一覧の見直しを定例にすることである。ここまでできると、バックアップは「あるらしいもの」から「説明できる運用」へ変わる。

第16章

# ヘルプデスク、依頼管理、ナレッジ整備

問い合わせには毎日答えている。パスワード再設定も、ソフト導入依頼も、PC 不調も、その場その場では処理している。だが一か月後を振り返ると、同じ質問にまた答えている。ソフト導入依頼では毎回同じことを聞き返している。FAQ は一応あるが古く、誰も見ない。チケットは件数を数えるだけで、何が繰り返されているかは見えていない。こういう状態は珍しくない。

ヘルプデスク運用が重くなるのは、依頼が多いからだけではない。もっと本質的には、依頼が標準化されず、自己解決へ寄せられず、対応の中で知識が残らないからである。すると、処理はしているのに、来月も同じことをする。

第2章では、窓口、記録、台帳、手順、優先順位という最小の土台を整えた。第16章では、その土台の上で、日々の依頼対応をどう軽くしていくかを見る。論点は、チケット化、依頼種別、申請フォーム、self-service、ナレッジである。

## 問い合わせは、必ずチケットや依頼記録に乗せる

第16章で最初に確認したいのは、すべての依頼が、最終的に一つの記録へ乗る状態になっているかである。ここで大切なのは、最初から完璧な ITSM ツールを入れることではない。共有窓口でも、簡易チケットでも、スプレッドシートでもよい。重要なのは、口頭、チャット、メールで来た依頼が、そのまま流れないことである。

ServiceNow が説明するように、service desk の tracking は、見落とし防止だけでなく、workload management と improvement のためのデータを生む。つまり、チケット化の目的は監視ではない。後から追えることと、後で直せることにある。

最低限、次は一つの場所で見たい。

- 受付日時
- 依頼者
- 依頼種別
- 優先度
- 状態
- 担当
- 完了日

第2章では入口を一つにすることを強調した。第16章で一段進めたいのは、受けた依頼が、例外なく記録へ乗る という運用である。チャットで直接頼まれても、その場で自分が代筆してよい。とにかく、後から「何が残っているか」が見えることが先である。

## 依頼種別と優先度を分ける

依頼管理が重くなる大きな理由の一つは、障害と依頼と変更が同じ流れで処理されることである。これでは、緊急障害に引っ張られて定型依頼が溜まり、定型依頼に時間を取られて重要障害の初動が遅れる。

Atlassian は、service request を separate workstream として扱うことを勧めている。service request とは、パスワード再設定、ソフトウェアライセンス、アクセス付与、新しい端末の手配、情報照会のような、繰り返し発生しやすい依頼である。これらは incident と違い、標準化しやすい。

だから、少なくとも次は分けたい。

- 障害
- 依頼
- 変更

この三つを分けるだけでも、見方が変わる。障害は復旧を急ぐ。依頼は必要情報を揃えて標準処理へ寄せる。変更は影響確認と承認が重要になる。全部を一つの箱に入れると、どの順で見るべきかが曖昧になる。

service request の中でも、低リスクで定型的なものはさらに標準化しやすい。たとえば次のようなものだ。

- パスワード再設定
- ソフト導入依頼
- アカウント作成依頼
- 権限追加依頼
- 備品や端末の手配依頼

これらは毎回ゼロから判断しない方がよい。必要項目、承認者、処理順を決めておくと、ひとり情シスの頭の負担が減る。

## 申請フローは、確認事項を前倒しする仕組みである

依頼フォームや申請フローを嫌がられることがある。「手間が増える」「気軽に頼めない」という反応である。だが実際には逆で、フォームの価値は、担当者が後で聞き返す手間を前に寄せることにある。

Atlassian の request forms は、conditional sections や custom layouts を持てる。これは見栄えの話ではない。依頼内容に応じて、必要な質問だけを出せるという意味である。たとえば、ソフト導入依頼なら、次のような項目を最初から取れる。

- 利用目的
- 利用者
- 利用端末
- 予算
- 承認者
- 希望時期

ここが抜けると、ひとり情シスは毎回同じ質問を返すことになる。結果として、依頼者も待ち、担当者も疲れる。

申請フローで重要なのは、すべてを厳しくすることではない。むしろ、低リスクの定型依頼は簡単にし、高リスクや例外だけ重くする方がよい。たとえば、標準ソフトの追加インストールは簡単な申請で済ませ、未承認 SaaS や高権限付与だけ別承認にする。この分け方がないと、全部が面倒になり、結局裏口依頼が増える。

フォームは利用者を管理する道具ではない。確認漏れを減らし、処理を速くする道具である。

## self-service へ寄せられるものを見極める

ヘルプデスク運用を軽くするうえで、self-service は重要である。ただし、何でも自己解決に寄せればよいわけではない。向いているものと向いていないものを分けた方がよい。

ServiceNow が示すように、self-service は common issues を自分で解決できるようにし、複雑な依頼へ時間を戻すための仕組みである。ここで代表例になるのが、パスワード再設定である。Microsoft Entra の SSPR も、まず selected group で試してから広げるやり方を案内している。つまり、self-service は全社一斉でなく、限定導入から始めてよい。

self-service に向きやすいのは、次のようなものだ。

- パスワード再設定
- VPN や Wi-Fi の基本案内
- 標準ソフトの申請方法
- よくあるエラー時の初動確認
- 申請状況の確認

逆に、個別事情が強いもの、権限判断が重いもの、機密性が高いものは、人が見る方がよい。大切なのは、切り分けることである。

self-service を導入する時に避けたいのは、「記事を読んでから来てください」で終わることである。記事が古く、見つけにくく、手順も複雑なら、利用者は余計に困る。self-service は、見つけやすく、短く、今の環境で使えることが前提である。

## ナレッジは、対応の副産物として育てる

ナレッジが増えない最大の理由は、別仕事として積み上がるからである。問い合わせに答えた後で、落ち着いたら記事を書く。多くの場合、その「後」は来ない。

Atlassian の KCS は、agents should always search the knowledge base first とし、knowledge を capture、structure、reuse、improve していく loop を示している。これはひとり情シスに特に向いている。なぜなら、対応しながら少しずつ残す方が、後でまとめて書くより現実的だからだ。

実務では、次の流れにすると続きやすい。

1. まず既存記事を探す
2. あればそれを使い、足りなければ直す
3. なければ短く新しく残す
4. 次に同じ問い合わせが来たら、その記事を使う

ここで重要なのは、最初から立派な記事にしないことである。短い FAQ でもよい。見出しと手順だけでもよい。大切なのは、次の一件を楽にできることである。

ナレッジ化に向いている題材は、同じ説明を二回以上したものだ。たとえば、VPN 接続手順、社内プリンタの使い方、標準ソフトの申請方法、共有フォルダ申請時の注意点。このあたりは、一本記事があるだけでかなり軽くなる。

## チケットは、件数より傾向を見るために使う

チケット管理を件数報告だけで終わらせると、改善に結びつきにくい。第16章で見たいのは、どの依頼が繰り返されているか、どこで滞留しているか、どの申請で差し戻しが多いかである。

ServiceNow も、tracking provides valuable data that can be analyzed to identify patterns, trends, and areas for improvement としている。つまり、チケットの価値は、何件処理したかより、何を減らせるかを見ることにある。

月に一度でもよいので、次を眺めたい。

- 繰り返し多い依頼は何か
- 差し戻しが多い申請は何か
- 待ち時間が長い依頼は何か
- 記事化できる問い合わせは何か

ここを見ると、改善点が見える。パスワード再設定が多いなら self-service を考える。ソフト導入依頼の差し戻しが多いならフォームを直す。同じ PC 設定の質問が多いなら初期案内を見直す。こうして、依頼対応が改善活動へつながる。

## 小さな会社で現実的に回るやり方

すべての会社が、本格的な service catalog や高度な virtual agent を入れられるわけではない。小さな会社では、最小構成でも十分効果がある。

現実的には、次のくらいから始めるとよい。

- 共有窓口を一つにする

- 依頼をチケットまたは一覧表へ乗せる
- request type を三つか四つに絞る
- よくある依頼のフォームを一つ作る
- FAQ を五本だけ作る
- パスワード再設定を self-service 候補にする

ここで意図的に避けたいのは、依頼分類を細かくしすぎることと、ナレッジを完璧にしようとすることである。分類が増えすぎると誰も正しく選べない。記事を完璧にしようとすると誰も書かない。小さな会社ほど、単純な型の方が強い。

## 通常時の例と、崩れた時の例

通常時の例では、IT 窓口へ入った依頼は必ず一つの記録へ乗る。依頼種別は、障害、依頼、変更に分かれている。標準ソフト導入や権限追加は request form で必要情報を先に集める。パスワード再設定は selected group から self-service を始めており、FAQ と合わせて単純問い合わせを減らしている。一次回答で使った説明は短いナレッジ記事へ残し、次回から記事を案内する。月に一度、繰り返し多い依頼を見て、フォームや FAQ を直す。結果として、問い合わせ件数はあっても、同じ説明を何度もゼロから繰り返さなくなる。

崩れた時の例では、依頼はチャット、口頭、メールに散っている。障害と依頼が同じ箱で扱われ、急ぎではない定型依頼が割り込む。ソフト導入依頼では毎回「誰が使うのか」「何のためか」を聞き返す。パスワード再設定は全部人手で受け、FAQ は古く、誰も見ない。チケットは件数だけ数えて終わりで、繰り返し傾向は見ていない。結果として、毎日忙しいのに、来月も同じことで忙しい。

## 最低限ここまではやる

第16章の内容を一度にすべて整えなくてもよい。だが、次の四つは早めに持ちたい。

1. 依頼が必ず記録へ乗る状態を作る
2. service request と incident を分ける
3. よくある依頼の request form を一つ作る
4. 繰り返し問い合わせを一件ナレッジ化する

この四つがあるだけで、ヘルプデスク運用はかなり軽くなる。ヘルプデスク運用とは、来た依頼に答えることではない。依頼を標準化し、単純なものを減らし、残った重要案件へ時間を使える状態を作ることである。

## 今日、今週、後でやること

今日やることは、直近一か月で多かった依頼を五つ書き出すことである。そこに、同じ説明を繰り返したものがあれば、ナレッジ候補である。

今週やることは、その中から一つを選び、request form か FAQ のどちらかへ落とすことである。たとえばソフト導入依頼ならフォーム、VPN 接続案内なら FAQ が向く。

後でやることは、月次で recurring requests を見直し、self-service 候補と記事更新候補を一つずつ決めることである。ここまでできると、ヘルプデスクは受け身の仕事から、軽くなる仕事へ変わる。

第IV部

# セキュリティ、法務、監査

事故を減らし、説明できる運用にする。

第17章〜第20章

第17章

# セキュリティ基礎対策

セキュリティ事故というと、高度な標的型攻撃や巧妙なゼロデイを思い浮かべやすい。だが、ひとり情シスや中小規模組織の現場では、もっと基本的な抜けから事故が始まることが多い。管理者アカウントに MFA がない。古い認証方式が残っている。怪しいメールが来ても報告先が曖昧で、そのまま開かれる。社内 Wi-Fi とゲスト Wi-Fi が同じで、複合機や私物端末も同じ場所にいる。送金依頼や口座変更依頼を、メール一本で確定してしまう。こうしたことは、珍しい失敗ではない。

第17章で扱うのは、ここを先に整えるための基礎対策である。高度な製品比較ではない。小さな会社でも優先しやすく、効果が出やすい最低線を扱う。中心になるのは、認証、メール、端末、ネットワーク、そして業務手順である。

## セキュリティ対策の全体像は、入口、成立、拡大の三段で考える

セキュリティ対策を考える時に有効なのは、個別製品名から入らないことである。まずは、どこから入られるか、入られた時に成功しやすいか、成功した後にどこまで広がるかで整理した方が分かりやすい。

第17章では、基礎対策を次の三段で考えたい。

- 侵入口を減らす
- 攻撃を成立しにくくする
- 被害面積を広げにくくする

侵入口を減らすとは、旧式認証を止める、怪しいメールを届きにくくする、管理されていない端末を減らす、といった対策である。攻撃を成立しにくくするとは、MFA、標準ユーザー運用、メール保護、端末保護を整えることだ。被害面積を広げにくくするとは、最小権限、ネットワーク分離、管理者権限の分離、送金確認手順のようなものを指す。

この順番が重要である。ログの保持や復旧は後続章で詳しく扱うが、第17章ではまず、そもそも起こりにくくする ことへ集中したい。

## マルウェア、フィッシング、BEC を分けて考える

セキュリティ対策がぼやけるのは、脅威を全部ひとまとめにしてしまうからである。少なくとも、マルウェア、フィッシング、BEC は分けて考えた方がよい。

マルウェアは、端末やサーバーで不正な動作をさせるものだ。添付ファイル、悪意あるリンク、脆弱なソフト、USB 媒体など入口は複数ある。ここでは、端末保護、更新、実行制御、権限管理が効く。

フィッシングは、利用者をだまして認証情報や承認操作を引き出す攻撃である。メール、SMS、チャット、音声通話の形を取ることもある。ここでは、メール保護、MFA、利用者教育、報告導線が効く。

BEC は business email compromise の略で、メールアカウントの乗っ取りやなりすましを通じて、送金、口座変更、請求先変更、機微情報送付をだまし取る詐欺である。FBI IC3 の 2024 年 9 月 11 日の PSA では、2013 年 10 月から 2023 年 12 月までの exposed dollar loss が 554 億ドル超とされている。ここでは、メール保護だけでなく、業務手順が重要になる。

この三つは似て見えるが、対策の置き場所が違う。だから、セキュリティ教育をやる だけでは足りないし、逆に 製品を入れる だけでも足りない。

## 端末の基礎防御を切らさない

端末は最も身近な入口である。ここでの基礎対策は、豪華な機能を盛ることではなく、最低線を切らさないことだ。

最低限、次はそろえたい。

- サポート中の OS と主要ソフトを使う
- 組み込みまたは管理されたウイルス対策を有効に保つ
- ローカル firewall を切らない
- 日常利用は標準ユーザーで行う
- 画面ロックを入れる
- 管理対象外の端末を増やさない

Microsoft は、Microsoft Defender Antivirus を Windows に built in の保護機能として位置づけており、disabled や uninstalled は一般に推奨していない。ここから見ても、EDR までは入れられない組織であっても、少なくとも組み込みのウイルス対策を無効化せず、更新され、状態確認できることが最低線になる。

ひとり情シスの現場では、専用製品がないから十分に守れない と考えやすい。だが実際には、何もないより、組み込み保護を正常に保ち、標準ユーザーで使い、怪しい挙動を報告できる方がはるかに強い。

ここで避けたいのは、例外端末の放置である。古い業務ソフトのために更新が止まった PC、個人判断でウイルス対策を切った PC、ローカル管理者のまま使われている　PC。こうした端末は、全体の防御線を静かに崩す。詳細なパッチ運用は第18章で扱うが、第17章の段階でも、守れていない端末を放置しない という姿勢は必要である。

## メールと認証の基礎防御を整える

最も効果が高い基礎対策の一つが、認証の近代化である。古い認証経路や password only の抜け道を残したまま MFA だけ議論しても、守りは弱い。

Microsoft Entra の security defaults は、全ユーザーの MFA 登録、管理者の MFA、legacy authentication protocols の block を基礎線としている。2024 年 7 月 29 日以降は、MFA 登録の 14 日猶予も廃止されている。Google Workspace でも、2025 年 5 月 1 日から username と password だけで接続する less secure apps をサポートしなくなった。両者に共通するのは、古い抜け道を残さない という考え方である。

実務では、次の順で進めるとよい。

- まず管理者アカウントに MFA を強制する
- 次に一般利用者へ広げる
- legacy authentication や less secure apps を止める
- 例外が必要な機器や古いクライアントを洗い出す

ここで大切なのは、MFA を入れることと、MFA を避けられる旧式認証を止めることを一体で見ることである。片方だけだと抜ける。

メール保護も同じで、教育だけでは不十分である。Microsoft 365 では、all cloud mailboxes に basic anti-phishing features があり、spoof intelligence や unauthenticated sender indicators のような保護がある。こうした既定の保護設定を確認せず、利用者の注意力だけに依存するのは危うい。

また、自社が受けるメールだけでなく、自社ドメインから出るメールのなりすまし対策も重要である。Google は DKIM に加えて SPF と DMARC の設定を推奨しており、DMARC はいきなり厳格にせず、まず monitoring してから段階的に強める流れを案内している。ここから分かるのは、送信ドメイン認証も基礎対策の一部だということだ。

実務で最低限見たいのは、次である。

- 管理者 MFA
- 全社 MFA の計画
- 旧式認証の停止
- anti-phishing や anti-spoofing の有効化
- 自社送信ドメインの SPF、DKIM、DMARC の状態確認

## ネットワークと境界は、社内だから安全と思わない

ゼロトラストという言葉は難しく見えるが、第17章の段階では単純に考えてよい。NIST SP 800-207 が言う通り、network location に基づく implicit trust を置かないことが本質である。つまり、社内 LAN にいるだけで安全だと思わないことである。

小さな会社でまず効くのは、次のような対策だ。

- guest Wi-Fi を社内用と分ける
- 複合機、カメラ、会議室機器を同じ場所に置きっぱなしにしない
- ルーター、AP、UTM の初期パスワードを残さない
- 管理画面をインターネットへさらさない
- VPN や管理画面へは MFA を使う

第11章でネットワーク構成は詳しく扱ったが、第17章での要点は、境界を信用しすぎない ことである。社内だから自由に見える、社内だから管理画面を開けてよい、社内だから高権限で使ってよい。こうした前提を少しずつ崩すだけで、防御はかなり安定する。

## 最小権限と業務手順で被害を小さくする

基礎対策というと、設定の話に寄りがちだ。だが、実害を小さくするには、設定だけでなく業務手順が必要である。特に BEC はその典型だ。

FBI IC3 は、口座情報変更の依頼に対して secondary channels や 2FA で verify することを prevention tips に入れている。これはとても実務的である。送金依頼や口座変更依頼がメールで来ても、そのメールだけで確定しない。既知の電話番号へかけ直す。別チャネルで確認する。承認者を一段増やす。このような確認統制が必要である。

最小権限も、ここで効く。もしアカウントが侵害されても、何でもできる権限でなければ被害面積は小さくなる。第8章で詳しく扱った通り、管理者権限の分離、役割ごとの権限、共有アカウントの抑制は、セキュリティ基礎対策でも中心にある。

もう一つ大切なのが、報告導線である。怪しいメールを受けた時、クリックしてしまった時、見慣れない承認通知が来た時、どこへ連絡すればよいかが曖昧だと、初動は遅れる。ここでは完璧な CSIRT は要らない。この窓口へ連絡する この件名で送る まずネットワークを切る のような最低線があるだけでよい。

利用者教育も、この文脈で書く方がよい。教育の目的は、全員をセキュリティ専門家にすることではない。怪しいものに気付き、止まり、報告できるようにすることである。責める文化より、早く知らせる文化の方が強い。

## 小規模組織で優先すべき対策

時間も予算も限られるなら、全部を同時にやろうとしない方がよい。第17章での現実的な優先順位は、次のようになる。

1. 管理者アカウントに MFA を強制する
2. 一般利用者へ MFA を広げる
3. legacy authentication や less secure apps を止める
4. 既定の anti-phishing と anti-spoofing を確認する
5. built-in antivirus と firewall が有効な端末だけを使う
6. guest Wi-Fi と社内ネットワークを分ける
7. 送金、口座変更、請求先変更は二経路確認にする

この順序には理由がある。最初の三つは、認証突破の入口をかなり減らす。次の二つは、よくあるメール起点と端末起点の侵入口を減らす。最後の二つは、侵入やなりすましがあっても被害が広がりにくくする。

ここで意図的に避けたいのは、高度な仕組みがないから後回し という考え方である。EDR、CASB、ZTNA のような製品があれば強い場面はある。だが、その前に MFA も guest 分離も二経路確認もないなら、優先順が逆になる。

## 通常時の例と、崩れた時の例

通常時の例では、管理者アカウントに先に MFA がかかっており、一般利用者も段階的に MFA へ移行している。古い認証方式は棚卸しされ、Google Workspace や Microsoft 365 のメール保護設定も確認済みである。自社ドメインには SPF、DKIM、DMARC の考え方が入り、まず監視しながら段階導入している。Windows

端末では built-in のウイルス対策と firewall が有効で、ローカル管理者での常用は避けている。guest Wi-Fi は社内と分かれ、送金や口座変更依頼は既知の電話番号で折り返し確認する。怪しいメールを見た時の報告先も決まっている。結果として、攻撃が絶対になくなるわけではないが、よくある入口で事故が起きにくい。

崩れた時の例では、管理者アカウントに MFA がなく、一般利用者にも password only の経路が残っている。古いメールクライアントや機器のために旧式認証が放置され、怪しいメール対策は利用者注意だけで済ませている。自社ドメインの送信認証は整っておらず、guest Wi-Fi と社内ネットワークも同じである。ある日、経理担当に役員名義の送金依頼メールが届き、そのまま処理される。後で調べると、差出人表示は似ていたが別ドメインで、しかも送金依頼を確認する二経路ルールもなかった。問題は、一つの製品が足りなかったことではなく、基礎対策の最低線がなかったことである。

## 最低限ここまではやる

第17章の内容を一度にすべて整えなくてもよい。だが、次の五つは早めに持ちたい。

1. 管理者アカウントに MFA を強制する
2. legacy authentication や less secure apps の残りを確認する
3. anti-phishing と anti-spoofing の既定保護を確認する
4. built-in antivirus と firewall を無効化していない端末だけを使う
5. 送金や口座変更はメール一本で確定しない

この五つがあるだけで、セキュリティ基礎対策はかなり前進する。セキュリティ基礎対策とは、高価な仕組みを足すことではない。よくある侵入口を減らし、認証を近代化し、被害が広がりにくい最低線を作ることである。

## 今日、今週、後でやること

今日やることは、管理者アカウントに MFA が強制されているかを確認し、同時に旧式認証や password only の例外が残っていないかを一つ洗うことである。

今週やることは、メール保護設定を見直し、受信側の anti-phishing 設定と、自社送信ドメインの SPF、DKIM、DMARC の現状を一覧にすることである。あわせて、送金や口座変更の二経路確認ルールを文書化したい。

後でやることは、guest Wi-Fi 分離、管理画面の保護、報告導線の明確化のような 社内だから安全 前提を崩す整備を順に進めることである。ここまでできると、セキュリティは不安の話から、優先順位を持って管理する実務へ変わる。

第18章

# 脆弱性管理、パッチ管理、構成管理

脆弱性情報は毎日のように出る。ブラウザ、OS、VPN　装置、業務アプリ、ネットワーク機器。重大と書かれた記事を見るたびに不安になる。だが、実務で本当に必要なのは、ニュースをたくさん追うことではない。その情報が自社に関係あるかを判断し、優先順位を決め、当て、確認し、当てられないものは別の方法で守ることである。ここができていないと、毎回騒いで終わる。

第18章で扱うのは、この判断ループである。脆弱性管理とパッチ管理と構成管理は、別々の仕事に見えて実際にはつながっている。何を持っているかを知り、どこに危険があるかを知り、どこから直し、どこを標準へ戻すかを決める。これを定常運用へ変えるのが第18章の目的である。

## 脆弱性管理は、収集、判断、適用、確認のループである

まず置きたいのは、脆弱性対応の骨格である。NIST　SP　800-40　Rev.4　は、enterprise patch management を identifying、prioritizing、acquiring、installing、verifying の流れで整理している。つまり、当てることだけが仕事ではない。

第18章では、次の流れで考えると分かりやすい。

- 情報を集める
- 自社影響を判断する
- 優先順位を付ける
- 適用または代替策を取る

- 適用結果を確認する
- 保留理由や例外を記録する

この中で特に抜けやすいのが、判断　と　確認　である。判断がないと、重大ニュースに引きずられやすい。確認がないと、配ったつもり、直したつもりで終わる。脆弱性管理とは、CVE を見た回数ではなく、このループが回っているかで決まる。

## 情報収集元を決める

脆弱性対応が崩れやすいのは、まず情報源が定まっていないからである。検索して見つける運用では遅い。少なくとも、見る場所の役割を分けておきたい。

最優先になるのは、製品ベンダーの advisory である。affected versions、fixed versions、workaround、回避策、再起動要否。こうした一次情報は、たいてい vendor advisory にある。自社が使っている VPN、OS、ブラウザ、メール基盤、認証基盤、主要 SaaS の advisory 受信先は最初に決めておいた方がよい。

日本語の集約源として有用なのが、JVN と JVN iPedia である。JVN iPedia は、JVN、国内製品開発者、NVD の脆弱性対策情報を日本語で集約している。さらに MyJVN は、脆弱性対策情報の効率的収集や、利用者 PC 上のソフトウェア製品バージョン確認を支援する仕組みを提供している。ひとり情シスにとっては、英語の vendor advisory だけで追うより現実的である。

そのうえで、active exploitation の優先判断に役立つのが CISA の Known Exploited Vulnerabilities、いわゆる KEV である。CISA は KEV を、実際に exploited in the wild であることが確認された脆弱性の authoritative source として維持し、民間を含むすべての組織に remediation prioritization への利用を勧めている。つまり、理論上危険 と すでに狙われている は分けて扱うべきだということである。

NVD も有用だが、役割は少し違う。CVSS や公開メタデータを見るための補助線と考えた方がよい。

ここで忘れてはいけないのが、自社の製品名とバージョンの一覧である。何を見ればよいか決まっていても、自社のどこにその製品があるか分からなければ判断できない。脆弱性管理は、情報収集だけで成立しない。資産とバージョンが結び付いて初めて動く。

## 優先順位は CVSS だけで決めない

NVD は、CVSS について `severity` を測る仕組みであり、`risk` そのものではないと明示している。ここは第18章の重要点である。CVSS 8.8 だから即日対応、6.5 だから来月でよい、と機械的には決められない。

優先順位付けでは、少なくとも次を重ねたい。

- 自社がその製品とバージョンを使っているか
- インターネット公開されているか
- 認証基盤、VPN、メール、公開 Web のような入口か
- 重要業務や機微データに近いか
- active exploitation が確認されているか
- workaround や代替策があるか
- すぐ当てると業務影響が大きいか

NIST SP 1800-31 も、security と mission impact を risk-based methodology で balance すると整理している。これは実務的である。脆弱性対応は、深刻度だけでなく、自社影響と業務影響の両方を見る必要がある。

たとえば、社内だけで使う端末向けの中程度脆弱性と、インターネット公開中の　VPN　装置にある高めの脆弱性では、後者を先に見る方が自然である。さらに、その脆弱性が KEV に載っているなら、優先度はもう一段上がる。

逆に、CVSS　　が高くても、自社でその機能を使っていない、公開もしていない、代替策もすでに入っているなら、急ぎ方は変わる。大切なのは、点数を見るではなく 自社に当てる ことである。

## 通常更新と緊急更新を分ける

NIST SP 1800-31 は、routine and emergency patching situations を分けて扱っている。これはそのまま実務に使える。通常更新と緊急更新は、同じレーンで回さない方がよい。

通常更新では、次のような流れを取りやすい。

- 月次または週次の更新日を決める
- test、pilot、production の順で段階展開する
- 再起動や業務影響の時間帯を調整する
- 失敗端末を追う
- 適用結果を確認する

Microsoft Intune の update rings も、test、pilot、production の deployment stages を作る考え方を取っている。deferral、restart、deadline を変えながら段階展開でき、必要なら　Pause、Resume、Uninstall　もできる。製品名はさておき、考え方としては、小さな会社でも参考になる。重要なのは、一斉配布一発勝負にしないことだ。

一方、緊急更新は別である。active exploitation が確認され、通常の deferral を待つ余裕がない時は、緊急レーンへ乗せる必要がある。Microsoft の expedited updates も、そのために deferral をバイパスして特定の security update を急いで入れる仕組みとして整理されている。ここから分かるのは、緊急更新は例外ではなく、運用として事前に用意しておくべきだということだ。

通常更新と緊急更新を分けないと、毎回すべてが緊急になり、結局どちらも崩れる。

## 更新できない資産には代替策を置く

ここは非常に重要である。更新できない資産は必ず出る。古い業務ソフト、依存関係が深い装置、停止できないサーバー、ベンダー都合で fix が遅い機器。問題は、当てられないこと自体より、当てられないまま何も足さないことだ。

NIST SP 1800-31 は、patching の代替として isolation methods や other mitigations を挙げている。つまり、パッチ不能は 終わり ではなく 別の守りへ切り替える合図 である。

現実的な代替策は次のようなものだ。

- 公開停止する
- 該当機能を止める
- ベンダーの workaround を適用する
- 外部から届かないようにする
- 別セグメントへ分離する
- 到達元を限定する
- 監視を強める
- 置き換え期限を決める

Fortinet の 2026 年 2 月 10 日の PSIRT では、Agentless VPN / FSSO の認証回避脆弱性に対し、affected versions、fixed versions に加えて、Disable unauthenticated bind on the LDAP server という workaround を示している。こうした advisory は実務的である。すぐ更新できないなら、公開を止めるか、回避策を入れるか、到達経路を絞るかを決めなければならない。

ここで避けたいのは、今は当てられないので保留　で止めることだ。保留なら、理由、代替策、期限、担当を残す必要がある。

## 構成管理は、標準と逸脱を管理する

脆弱性管理と構成管理は別物に見えるが、実際には近い。標準構成から逸脱していると、同じ脆弱性でも被害が大きくなりやすいし、更新も当たりにくくなる。

NIST の National Checklist Program は、security configuration checklist を、hardening guide や benchmark を含むものとして位置づけ、configured properly の確認や unauthorized changes の特定にも使えるとしている。Microsoft の security baselines も、Microsoft-recommended configuration settings をまとめたもので、Group Policy、Configuration Manager、Intune などで適用できるとしている。つまり、構成管理とは 最初に固める だけでなく 今もその状態かを見る ことでもある。

第18章で扱いたい構成管理は、次の流れである。

- 標準構成を決める
- どの資産に適用するかを決める
- 例外設定を記録する
- 逸脱を見つける
- 標準へ戻すか、例外として残すかを判断する

ここで重要なのは、標準をゼロから作り込まないことである。Microsoft も industry-standard configuration として well-tested な baseline を使うことを勧めている。小さな会社ほど、既存の baseline や checklist を起点にした方が速く、ぶれにくい。

また、構成逸脱は静かに積み上がる。ある PC だけ firewall が切られている。あるサーバーだけ古い TLS 設定が残っている。ある端末だけローカル管理者のまま使われている。これをそのままにすると、脆弱性管理をいくら回しても、足元から崩れる。

## 小さな会社で現実的に回るやり方

全部の製品を同じ密度で追うのは現実的ではない。小さな会社では、重要資産から回す方がよい。

現実的には、次のくらいから始めるとよい。

- 重要資産を上位 10 件から 20 件に絞る
- それぞれの製品名とバージョンを一覧にする
- 主要製品の advisory 受信先を決める
- 月次更新日を決める
- 緊急更新条件を決める
- 更新失敗端末と保留資産の一覧を持つ
- baseline を一つ決めて準拠確認を始める

緊急更新条件は、たとえば次のように単純でよい。

- KEV 掲載
- active exploitation が確認されている

- インターネット公開資産が対象
- 認証、VPN、メール、公開 Web の入口に関わる

この条件に当たるものは、通常月次を待たずに判断会議を入れる。そこまで大げさでなくても、同日中に影響確認を始める、公開停止判断をする、回避策を検討する、といった最低線は持ちたい。

## 通常時の例と、崩れた時の例

通常時の例では、ひとり情シスがまず主要資産一覧を持っている。OS、ブラウザ、VPN 装置、認証基盤、公開サーバー、主要業務アプリの製品名とバージョンが分かる。vendor advisory と JVN / JVN iPedia を普段から見ており、KEV 掲載の有無も確認する。CVSS　は参考にするが、そのまま順位にはせず、公開有無、入口性、業務影響を重ねて判断する。月次更新は test、pilot、production で段階展開し、失敗端末は一覧で追う。active exploitation が確認された時だけ緊急レーンへ切り替え、必要なら公開停止や workaround を先に入れる。標準構成から外れた端末や機器も定期に見直し、標準へ戻す。結果として、ニュースに反応するのでなく、運用として回る。

崩れた時の例では、脆弱性情報は X やニュースで知るだけで、どの製品を自社が使っているかはすぐ出てこない。CVSS　が高いものだけを追っているが、インターネット公開中の認証機器は後回しになる。月次更新は一斉配布で、失敗端末は把握していない。fix が出ない機器もそのまま公開し続ける。標準設定からの逸脱も見ていない。ある日、VPN 装置の advisory が出るが、影響確認が遅れ、暫定対策も打たないまま数日が過ぎる。問題はニュースの見落としではなく、定常運用がなかったことである。

## 最低限ここまではやる

第18章の内容を一度にすべて整えなくてもよい。だが、次の五つは早めに持ちたい。

1. 重要資産の製品名とバージョンを一覧にする
2. vendor advisory、JVN、KEV の見る先を決める
3. CVSS 以外の優先判断軸を持つ
4. 通常更新と緊急更新のレーンを分ける
5. 更新できない資産の代替策と期限を記録する

この五つがあるだけで、脆弱性管理はかなり実務的になる。脆弱性管理とは、CVE を眺めることではない。自社に関係あるものを見極め、優先順位を付け、当て、当てられないものは代替策で囲い、標準構成へ戻し続ける運用である。

## 今日、今週、後でやること

今日やることは、公開資産と認証系資産を中心に、重要資産を 10 件だけ書き出し、製品名とバージョンが分かるかを確認することである。分からないものは、それだけで脆弱性管理上の宿題になる。

今週やることは、その一覧に対して advisory の受信先を決め、通常更新日と緊急更新条件を一枚にまとめることである。あわせて、更新不能資産があれば、保留理由と代替策を書きたい。

後でやることは、標準構成を一つ決め、逸脱一覧を持ち、更新失敗端末や例外資産を月次で見直すことである。ここまでできると、脆弱性対応は不安ベースの作業から、優先順位を持って回る保守運用へ変わる。

第19章

# 個人情報保護、社内規程、監査対応

個人情報保護の話になると、多くの会社はすぐに規程や同意文の話を始める。もちろんそれも必要である。だが、ひとり情シスの実務で本当に詰まるのはそこではない。どの業務が個人情報を扱っているのか、どこに保存されているのか、誰が触れるのか、外へどう出ていくのか、いま残っている記録で説明できるのか。ここが曖昧なままでは、規程を何本作っても回らない。

個人情報保護委員会のガイドラインや Q&A は、法律の解釈だけでなく、かなり実務的なヒントを含んでいる。責任者を置くこと、取扱規律を決めること、アクセス制御や教育を行うこと、取扱状況を確認できる記録を残すこと、委託と第三者提供を切り分けること。第19章では、これらを 日常運用へどう落とすか に絞って整理する。

## 個人情報、個人データ、保有個人データを分けて考える

最初に押さえたいのは、似た言葉を混ぜないことである。個人情報保護委員会の通則編は、個人情報 個人データ 保有個人データ を明確に使い分けている。ここが曖昧だと、必要な義務も曖昧になる。

ざっくり整理すると、次のように考えると実務へ落としやすい。

- 個人情報は、特定の個人を識別できる情報全般である
- 個人データは、個人情報データベース等を構成する、管理対象として扱っている個人情報である

- 保有個人データは、自社が開示、訂正、利用停止などに応じる権限を持つ個人データである

実務では、この違いをこう置き換えると分かりやすい。採用応募書類、名刺、問い合わせメール、従業員情報、顧客管理システムのレコード。これらのうち、日常業務で検索し、更新し、出力し、保持しているものが 個人データ になる。さらに、その中で自社が本人からの請求に対応する主体であるものが 保有個人データ になる。

つまり、個人情報保護の第一歩は、用語を覚えることではなく、次を言えるようにすることである。

- どの業務が個人情報を扱っているか
- どのシステムや保存場所にあるか
- どの項目を持っているか
- 利用目的は何か
- 誰がアクセスできるか
- 外部委託先や外部提供先があるか
- いつまで持ち、どう消すか

この洗い出しがないと、個人情報保護はいつまでたっても 総論 のままで終わる。監査で困るのも、漏えい時に困るのも、退職者データの削除で困るのも、たいていはここが曖昧だからである。

たとえば人事労務だけ見ても、入社書類、勤怠、給与、評価、健康診断、緊急連絡先など、個人情報は複数のシステムへ分かれている。営業なら、顧客管理、問い合わせフォーム、メール、共有フォルダ、名刺管理などに散らばる。まずはこの散らばりを見える化する必要がある。

なお、給与や社会保険の実務では、マイナンバーのように本章よりさらに強いルールがかかる情報もある。第19章は個人情報保護全般の土台を扱う章であり、そうした個別制度の詳細には深入りしないが、同じ個人情報でも追加ルールがある場合がある ことは意識しておきたい。

## 安全管理措置を、日常運用へ落とす

個人情報保護の話が空中戦になりやすい理由は、安全管理措置を講じる　という言葉が抽象的だからである。通則編は、単に強く守れと言っているのではない。基本方針、取扱規律、組織的、人的、物理的、技術的な措置へ落としている。ひとり情シスにとって大事なのは、この分け方をそのまま運用へ持ち込むことだ。

まず土台になるのは、基本方針と取扱規律である。大げさな冊子である必要はない。どのデータを誰がどう扱うか、持ち出しや外部送付をどう管理するか、委託時に何を確認するか、保存期間と削除をどう決めるか。この骨格がないと、個別判断がすべて場当たりになる。

その上で、組織的安全管理措置は 体制と確認 の話である。責任者を決める。違反や兆候を見た時の報告先を決める。利用状況等の記録を残す。取扱状況を確認できる手段を持つ。定期的に自己点検や監査をする。個人情報保護委員会の通則編も、利用状況等の記録、取扱状況確認の手段、漏えい等対応体制、定期点検や監査を挙げている。ここで重要なのは、個人情報保護が 年一回の規程見直し ではなく、月次や四半期の確認に乗ることだ。

人的安全管理措置は 人が崩す部分を減らす 話である。教育、秘密保持、入社時の説明、異動時の権限見直し、退職時の失効。通則編も、従業者への周知徹底と適切な教育を求めている。第8章、第9章で扱った IAM やライフサイクル管理は、そのまま個人情報保護の実務でもある。

物理的安全管理措置は 持ち出し、置きっぱなし、捨て方 の話である。紙の申請書、印字帳票、USB メモリ、貸与 PC、廃棄前端末。個人情報保護では、紙も媒体もまだ現役である。施錠、持ち運び方法、廃棄時の復元不能化、責任者確認といった基本動作を甘く見ない方がよい。第10章の端末回収と退役、第15章の削除や廃棄の話は、ここへ直接つながる。

技術的安全管理措置は 誰が入れて、何ができて、どう記録されるか の話である。アクセス制御、識別認証、不正アクセスや不正ソフトウェアからの保護、漏えい等防止。通則編も、アクセス権の限定、ユーザー識別、外部からの不正アクセス防止、情報システム利用に伴う漏えい等防止を求めている。ここで第8章の権限管理、第17章の基礎対策、第18章の構成管理が個人情報保護とつながる。

大事なのは、これらを別々の棚に入れないことである。アクセス権棚卸し、管理者権限の分離、持ち出し制限、削除証明、アクセスログ、点検記録。これらはセキュリティ運用でもあり、個人情報保護法上の安全管理措置でもある。ひとり情シスは、この重なりを意識した方が仕事が増えにくい。

## 委託、第三者提供、クラウド利用を混同しない

第19章で最も実務的な論点がここである。外部へ個人データが出る場面は多い。給与計算、配送、サポート、クラウド、外部分析、親会社への報告、業務提携。ところが、この全部を 委託 と呼んだり、逆に全部を 第三者提供 と呼んだりすると、同意、監督、記録の判断を誤る。

まず、利用目的の達成に必要な範囲内で、個人データの取扱いを外部へ委ねる場合は 委託 である。個人情報保護委員会の FAQ でも、委託に伴う提供は第三者に該当せず、本人同意は不要と整理している。たとえば、給与計算を外部へ委託する、配送のために住所情報を配送業者へ渡す、データ入力を外部へ委ねる、といった場面である。

ただし、委託だから自由に使ってよいわけではない。通則編は、委託先は委託された業務の範囲内でのみ取り扱えると整理している。つまり、給与計算委託先が従業員データを自社分析に使う、配送業者が顧客データを営業に使う、といったことは別問題になる。第6章で扱ったように、委託先監督は必要である。

一方、相手先が自社の判断で利用するなら、第三者提供として扱う必要がある。営業提携先へ顧客一覧を渡す、親会社や関連会社へ名簿を渡して各社の判断で使う、イベント共催先へ参加者名簿を渡す。こうした場面は、委託とは違う。ここを 外部委託です と呼んで済ませると危ない。

混同を避けるためには、次の問いを置くとよい。

- 相手は自社の指示の範囲内だけで処理するのか
- 相手は自社の目的で利用するのか
- 本人へどう説明しているのか
- 記録や確認義務が生じるのか

もう一つ誤解が多いのがクラウド利用である。個人情報保護委員会の　Q7-53 は、クラウド利用が第三者提供や委託に当たるかを、事業者がその個人データを取り扱うこととなっているか で判断するとしている。つまり、クラウドだから自動的に第三者提供、クラウドだから自動的に委託、ではない。

たとえば、契約上、クラウド事業者が保存データを取り扱わないこととなっており、適切なアクセス制御もされている場合には、第三者提供でも委託でもない整理があり得る。逆に、事業者が保守や運用のために個人データを取り扱うなら、委託として見る必要が出てくる。ひとり情シスは、この違いを雑に流さない方がよい。

ここで大切なのは、分類だけでは終わらないことである。外部に出る場面では、承認経路、契約条件、利用目的、データ項目、保存期間、削除方法、再委託可否を確認したい。第19章は法的論点の章だが、実務としては 表に落とす のが最も効く。

また、第三者提供やその受領に関する記録は、専用の巨大台帳を新しく作ることだけが答えではない。個人情報保護委員会の Q&A は、既存の契約書などで記録事項を満たせば記録として認められ、転送ログも記録として認められるとしている。これは重要である。申請フォーム、承認記録、契約書、転送ログ、受領確認メール。これらを後で引けるように保存しておけば、日常運用の中で証跡を積み上げられる。

## 社内規程と利用ルールは、短く具体的に整える

規程は多ければよいわけではない。読まれず、守られず、監査でも説明に使えない規程は、実務上ほとんど価値がない。ひとり情シスが目指すべきなのは、短くても判断に使えるルールである。

最低限、次の内容は明確にしておきたい。

- どの種類の個人データを扱うか
- 利用目的は何か
- だれがアクセス権を申請し、承認し、棚卸しするか
- 外部送付や持ち出しの条件は何か
- 委託時に何を確認するか
- 保存期間と削除方法はどうするか
- 苦情や問い合わせの窓口はどこか
- 開示、訂正、利用停止等の請求にどう対応するか

特に　保有個人データ　は、本人の知り得る状態に置くべき事項がある。通則編は、事業者の氏名又は名称と住所、法人なら代表者氏名、利用目的、請求手続、苦情申出先、安全管理のために講じた措置などを周知対象としている。つまり、プライバシーポリシーや案内ページに 適切に管理します とだけ書いて終わりでは足りない。

ここでも、規程は抽象語を減らした方がよい。たとえば次の二つを比べると差が分かる。

- 悪い例
  - 個人情報は適切に管理する
- 良い例
  - 人事システムの閲覧権限は人事責任者承認で付与し、四半期ごとに棚卸しする

前者はきれいだが運用できない。後者は地味だが、申請、承認、棚卸し、証跡までつながる。ひとり情シスの規程は、こういう具体性を持っていた方が強い。

また、ルールは　個人情報保護だけの文書　に閉じ込めない方がよい。入社、異動、退職のフロー、端末配布ルール、共有フォルダの権限ルール、SaaS　導入申請、外部送付承認。これら既存運用に個人情報保護の観点を埋め込む方が、別紙を増やすより実効性が高い。

## 監査で見られる証跡を、平時から残す

監査対応でありがちな失敗は、規程ファイルだけを揃えて安心することだ。監査で本当に見られるのは、そのルールが実際に回っているか である。言い換えると、監査は書類の有無より、運用の痕跡を見る。

個人情報保護の監査で役に立つ証跡は、たとえば次のようなものである。

- 個人データを扱う業務とシステムの一覧
- データ項目、利用目的、保存場所、委託先の一覧
- アクセス権申請と承認の記録
- 権限棚卸しの記録
- 委託契約書や覚書
- 第三者提供時の承認記録
- データ転送ログ、受領確認、エクスポート記録
- 削除、廃棄、返却の証明
- 点検記録、監査記録、是正記録
- 教育実施記録

個人情報保護委員会の通則編も、取扱規律に従った運用の確認のために、システムログその他の記録や業務日誌の作成を通じて、個人データの取扱いを検証可能とすることが重要だとしている。さらに、取扱状況を確認するためには、データベースの種類や名称、項目、責任者、利用目的、アクセス権者などを明確にしておくことが考えられるとしている。これは、そのまま監査証跡の設計図になる。

つまり、監査証跡は監査のために別で作るより、日常運用の中で自然に残るようにした方がよい。SaaS　の申請フォームに利用目的と委託先を入れる。アクセス権申請に保存場所と取扱理由を入れる。削除時に実施日と確認者を残す。転送時に承認チケットとログを紐付ける。こうすると、監査前に慌てて記憶をたどる必要が減る。

また、証跡は　ある　だけでは弱い。どこにあるか、誰が取り出せるか、いつまで残すかが決まっていないと、監査当日に見つからない。証跡の保存場所を一つか二つに絞り、命名と保存期間を決めておきたい。

## 小さな会社で現実的に回るやり方

すべての個人情報を一気に洗い出し、全ルールを一斉更新し、全システムの証跡を整理するのは現実的ではない。小さな会社ほど、重要な業務から順に固めた方がよい。

現実的には、次の五つくらいから始めると回しやすい。

- 人事
- 勤怠
- 給与
- 顧客管理
- 問い合わせ対応

この五つについて、まず一枚の表を作る。列は多くなくてよい。

- 業務名
- システム名または保存場所
- 主なデータ項目
- 利用目的
- アクセスできる役割
- 委託先または外部連携先
- 保存期間
- 削除方法

この表ができるだけで、かなりの論点が見える。共有フォルダに昔の応募書類が残っている。給与データの委託契約が古い。問い合わせフォームの転送先が個人メールである。退職者のフォルダが削除されていない。どれも、表にしないと見えにくい。

次に、外部へ出る場面だけを色分けするとよい。

- 委託
- 第三者提供
- クラウド利用
- 社内のみ

この分類ができると、契約確認、承認経路、証跡の残し方が揃えやすくなる。さらに、月次で権限見直し、四半期で主要業務点検、年次で規程見直しといった最小サイクルを置けば、小さな会社でも十分回る。

## 通常時の例と、崩れた時の例

通常時の例では、人事、勤怠、給与、顧客管理、問い合わせ対応の五業務が一覧化されている。給与計算は外部委託として整理され、契約書、委託範囲、削除や返却条件が確認されている。顧客管理システムはアクセス権が役割ごとに分かれ、退職や異動のたびに見直される。問い合わせデータを外部ベンダーへ渡す場合は、委託か第三者提供かを事前に判断し、承認記録と転送記録を残す。プライバシー案内には、利用目的、苦情窓口、開示等請求手続、安全管理措置の概要が整理されている。監査で聞かれた時も、規程だけでなく、申請、承認、棚卸し、ログを出せる。

崩れた時の例では、個人情報保護規程はあるが、どの業務がどの個人データを持っているか一覧がない。給与委託先へ渡しているデータの範囲も曖昧で、契約書もすぐ出てこない。営業が提携先へ顧客一覧を渡しているが、委託と呼んで済ませており、実際には相手が自社目的で利用している。問い合わせフォームのデータは個人メールへ自動転送され、アクセス権棚卸しもしていない。監査で 誰がアクセスできるか どの承認で外へ出したか 削除した証跡はあるか と聞かれると答えに詰まる。問題は、法律を知らないことより、運用と証跡がないことである。

## 最低限ここまではやる

第19章の内容を一度に全部整えなくてもよい。だが、次の五つは早めに持ちたい。

1. 個人データを扱う主要業務とシステムを一覧にする
2. 各業務について、利用目的、アクセス権、外部委託先、保存期間、削除方法を決める
3. 外部へ出る場面を、委託、第三者提供、クラウド利用に分けて整理する
4. 本人向け周知事項と社内ルールを、短く具体的に整える
5. 承認記録、契約書、転送ログ、棚卸し記録を、後で引ける形で保存する

この五つがあるだけで、個人情報保護はかなり実務になる。個人情報保護とは、規程を置くことではない。どの業務でどの個人データを扱い、誰が触れ、どこへ渡し、どんな記録を残すかを説明できる状態を作ることである。

## 今日、今週、後でやること

今日やることは、人事、給与、勤怠、顧客管理、問い合わせ対応の五業務だけを取り上げ、どのシステムや保存場所に個人データがあるかを書き出すことである。ここで空欄になる項目が、そのままリスクになる。

今週やることは、その一覧に 利用目的 アクセスできる役割 委託先または外部提供先 保存期間 削除方法 を足すことである。あわせて、外部に出る場面を委託、第三者提供、クラウド利用へ分け、承認経路と記録の残し方を決めたい。

後でやることは、本人向け周知事項を見直し、権限棚卸し、削除証跡、点検記録の残し方を整えることである。ここまでできると、個人情報保護は 規程を持っている会社 から 運用を説明できる会社 へ変わる。

第20章

# 教育、訓練、委託先を含むセキュリティ運用

セキュリティ事故は、技術的な抜けだけで起きるわけではない。怪しいメールを開いてしまう。口座変更依頼をメールだけで信じてしまう。委託先が共有アカウントで接続している。取引先から届いたファイルを、そのまま開いてしまう。どれも、設定だけでは防ぎきれない。だから、セキュリティを本当に回すには、人と外部接点を含めた運用が必要になる。

第20章で扱うのは、その運用である。年1回の研修をやったかどうかではない。怪しい時に止まれるか、報告できるか、管理者や経理が自分向けのリスクを理解しているか、委託先にも最低限のルールが届いているか。ここまでできて初めて、セキュリティ教育は効き始める。

## 教育は、イベントではなく運用である

セキュリティ教育が形骸化する最大の理由は、受講したかどうか　をゴールにしてしまうことだ。NIST SP 800-50r1 は、learning program を life cycle approach で運用し、behavior change と security culture につなげる考え方を示している。つまり、教育は単発イベントではなく、繰り返し改善する運用として持つ方がよい。

第20章では、教育を次の五つに分けて考えたい。

- awareness
- training
- exercise
- notice

- review

awareness は、危険に気づくための土台である。training は、役割に応じて何を判断し何をするかを学ぶことだ。exercise は、実際に動けるかを試す。notice は、新しい手口や直近事例を短く共有する。review は、訓練や実際の出来事の後に、何を直すかを決める。これを全部ひっくるめて 教育 と見た方が、現場では回りやすい。

たとえば、eラーニングを年1回受けるだけでは、3か月後の不審メールに対応できるとは限らない。逆に、短い注意喚起が毎月あり、怪しいメール訓練の後に振り返りがあり、報告窓口が明確なら、現場は少しずつ止まれるようになる。教育の価値は、知識量より行動変化で見るべきである。

だから、第20章で見たい指標も受講完了率だけではない。どれだけ報告が来たか。訓練後に報告率が上がったか。新しい手口の共有ができているか。委託先を含む関係者へ同じルールが届いているか。NIST SP 800-50r1 も metrics と evaluation を重視しているが、小さな会社では複雑である必要はない。まずは 受けたか に加えて 報告したか 確認したか を見るだけでもよい。

## 標的型メール訓練と注意喚起を分けて回す

フィッシングや標的型メールへの対策は、訓練だけでは足りない。CISA は、training should be provided at regular intervals としつつ、between trainings の周知も勧めている。ここから分かるのは、訓練と日常の注意喚起は別物であり、両方必要だということだ。

日常の注意喚起では、次のような内容を短く回すとよい。

- 最近多い手口
- 自社や取引先で実際に起きた事例

- 怪しいメールを見た時の確認方法
- 報告先と報告方法

CISA　は、怪しい連絡を受けた時には、そのメールへ返信したり本文中の電話番号を使ったりせず、known contact method で確認するよう勧めている。これは重要である。ひとり情シスの現場では、メールの真偽を技術的に完全判定するより、怪しい時は既知の電話番号か既知の連絡先で確認する　という行動ルールを浸透させた方が強い。

標的型メール訓練では、さらに一歩進めて、実際の行動を確認したい。ただし、ここでよくある失敗が、クリック率だけを見て終わることである。NIST　の Phish Scale は、訓練メールの難易度を評価し、その難易度を踏まえて click rate と report rate を解釈する考え方を示している。つまり、難しい訓練でクリック率が高かったという事実だけでは、十分な評価にならない。

ここで見たいのは、少なくとも次の三つである。

- クリックしたか
- 報告したか
- その訓練はどれくらい難しかったか

この見方をすると、訓練の目的は クリックした人を見つけること ではなく、見抜けなかった時でもすぐ報告できる状態を作ること　に変わる。実務ではこちらの方が重要である。事故はゼロにできなくても、早く気づき、早く知らせれば被害は小さくできる。

また、訓練で扱う題材は、いわゆる怪しい添付ファイルだけでは足りない。IPA の BEC 事例集には、偽口座送金、口座変更依頼、社長なりすまし、取引先アカウント乗っ取りなどが並んでいる。つまり、訓練は　リンクを踏むか　だけでなく、送金や口座変更をどう確認するか まで広げた方が実務に効く。

## 管理者向け教育と現場向け教育を分ける

全社員へ同じ内容を一斉配信するだけでは、必要なことが抜ける。NIST SP 800-50r1 は diverse set of employee audiences を前提にしており、IPA の 5分でできる！情報セキュリティポイント学習 も 経営者・管理者向け と 従業員向け を分けている。つまり、役割別教育は大企業だけの話ではない。

最低限でも、次のように分けたい。

- 一般社員
  - 不審メール、外部共有、パスワード、MFA、報告先
- 新入社員
  - 会社の基本ルール、使ってよいツール、持ち出し、相談先
- 管理職
  - 例外承認、部下からの相談対応、事故時の連絡判断
- 経理
  - 送金依頼、口座変更、請求書差し替え、既知経路確認
- 人事
  - 個人情報、委託先共有、退職者データ、権限見直し
- 管理者権限保有者
  - 特権アカウント、リモート接続、ログ、設定変更、共有禁止
- 役員
  - なりすまし、急ぎ案件、機密資料、権限委任の線引き

ここで重要なのは、教育テーマを増やしすぎないことだ。一般社員に管理者向けの深い話をしても定着しないし、経理に一般的な phishing の話だけをしても BEC 対策には足りない。ひとり情シスは、誰が一番狙われやすく、どの判断を誤ると痛いか で内容を切る方がよい。

たとえば経理なら、怪しいリンクより先に 振込先口座の変更依頼をメール一本で確定しない を徹底した方が効果が高い。管理者なら、共有アカウントで作業しない 作業後に権限を戻す 外部ベンダーへ管理者権限を渡しっぱなしにしない が重要になる。教育は平等である必要はない。必要に応じて違っていてよい。

## 委託先、協力会社、取引先も、セキュリティ運用の対象である

自社社員だけ教育しても、委託先が弱ければ入口は残る。委託先が自社アカウントを使う、共有フォルダへ入る、ファイル授受をする、リモート接続するなら、その相手はセキュリティ運用の対象である。

ここで大事なのは、契約条項だけで終わらせないことだ。第6章では責任分担や契約の話をしたが、第20章で扱いたいのは、日常運用として何を守ってもらうかである。最低限、次は決めたい。

- 外部用アカウントを個人単位で発行する
- MFA を必須にする
- 接続元や利用時間を必要範囲に絞る
- 承認されたツール以外でファイル授受しない
- 共有アカウントを使わない
- 作業内容と連絡先を明確にする
- 契約終了時の停止、返却、削除を決める

これらは難しい統制ではない。だが、曖昧なままだと事故が起きやすい。委託先が自分の私用クラウドでファイルを受け取る。誰が接続したか分からない共有 ID を使う。障害時の連絡先が営業担当しか分からない。こうした状態では、技術的に守っていても崩れやすい。

また、委託先が自社システムを触るなら、自社社員と同じ注意喚起が必要な場面もある。たとえば、今流行している偽ログイン画面の注意、ファイル共有方法の変更、緊急時の報告先などである。外部だから一切共有しないのではなく、必要なルールと連絡は渡す必要がある。

## サプライチェーン全体で考える防御

取引先やベンダーのリスクを見るというと、大企業向けの重い監査を想像しやすい。だが、NIST SP 1305 も CISA の SMB 向け Vendor SCRM 資料も、もっと実務的である。中心は `supplier requirements を定義し、伝え、確認する` ことだ。

ひとり情シスの現場で、まず決めたい要求は次のようなものだ。

- 自社データをどこで扱うか
- 誰がアクセスできるか
- MFA やアクセス制御をどうするか
- インシデント時にいつ誰へ連絡するか
- 再委託の条件は何か
- 契約終了時にデータをどう返却、削除するか

ここでのポイントは、相手を漠然と信用するかどうかではない。何を求めるかを明文化し、それが守られているかを確認することである。NIST SP 1305 は、organization が smart acquirer and supplier になるために、supplier requirements

を定義し communicate することを強調している。CISA の SMB 向けテンプレートも、購入や委託の前に、ベンダーの reporting and vetting processes を確認するための質問票を用意している。

つまり、サプライチェーン防御は 契約書があるから安心 でも 有名ベンダーだから安心 でもない。要求、確認、連絡経路の三つが必要である。

また、日本でも取引先からのセキュリティ要求は現実のものになっている。IPA の 2025年5月27日公表の調査では、1割強の企業が取引先から情報セキュリティ対策の要請を受けている とされている。要請内容として 秘密保持のための措置 も多い。これは、第20章が社内教育の章であると同時に、取引の信頼を守る章でもあることを示している。

## 小さな会社で現実的に回るやり方

小さな会社で最初から大規模な教育プログラムを作る必要はない。むしろ、長い研修ほど続かない。現実的には、次のくらいから始めるとよい。

- 入社時に 30 分だけ基本ルールを説明する
- 月1回、5分で読める注意喚起を流す
- 四半期ごとに短い標的型メール訓練を行う
- 訓練後に なぜ見抜けなかったか を短く共有する
- 経理、人事、管理者向けに別の注意喚起を足す
- 委託先向けに一枚の接続、共有、報告ルールを渡す

IPA の 5分でできる！情報セキュリティポイント学習 のように、短く学べる無料教材もある。こうしたものを土台にして、自社で必要な話だけ足す方が現実的である。

また、教育は やった で終わらせず、実際に何が起きたかへつなげたい。怪しいメールが来た。取引先から突然共有リンクが届いた。ベンダーが新しい接続方法を求めてきた。こうした出来事を材料にして短く振り返る方が、抽象的な研修より定着しやすい。

## 通常時の例と、崩れた時の例

通常時の例では、新入社員は入社時に基本ルールと報告先を教わる。一般社員には毎月短い注意喚起が流れ、四半期ごとに標的型メール訓練を行う。訓練結果はクリック率だけでなく報告率も見て、次回の題材を調整する。経理には口座変更依頼と送金依頼の確認ルールを別に周知し、管理者権限保有者には特権アカウント運用の注意を出す。委託先には個別アカウントと MFA を必須にし、ファイル授受と緊急連絡先を一枚にまとめて渡す。こうすると、教育、注意喚起、委託先ルールが一本線でつながる。

崩れた時の例では、年1回の　eラーニングは実施しているが、報告先を誰も覚えていない。標的型メール訓練はクリック者の一覧を出して終わりで、振り返りもない。経理も人事も一般社員と同じ教材だけを受けている。委託先は共用アカウントで接続し、ファイルは各自のやりやすい方法で受け渡している。ある日、取引先担当者になりすました口座変更依頼メールが届き、経理がそのまま処理する。後で見ると、確認電話もせず、報告窓口も使われていない。問題は 教育をしていなかった ことではなく、教育が運用へつながっていなかったことである。

## 最低限ここまではやる

第20章の内容を一度に全部整えなくてもよい。だが、次の五つは早めに持ちたい。

1. 怪しいメールや不審な依頼の報告先を明文化する

2. 年1回以外の短い注意喚起を定期化する
3. 標的型メール訓練では報告率も見る
4. 経理、人事、管理者向けの教育内容を分ける
5. 委託先向けに、アカウント、接続、共有、報告の最低限ルールを一枚にする

この五つがあるだけで、セキュリティ教育はかなり実務になる。セキュリティ教育とは、研修を実施することではない。社員も委託先も怪しい時に止まり、既知の経路で確認し、すぐ報告できる運用を作ることである。

## 今日、今週、後でやること

今日やることは、不審メール、不審な送金依頼、不審な外部接続依頼が来た時の報告先と確認経路を、一枚で書き出すことである。ここが曖昧なら、教育しても動けない。

今週やることは、対象者を一般社員、経理、人事、管理職、管理者権限保有者に分け、それぞれに何を教えるかを決めることである。あわせて、委託先向けの接続、共有、報告ルールを短く文書化したい。

後でやることは、標的型メール訓練の題材と結果の見方を整え、報告率や振り返りまで含めて回すことである。ここまでできると、セキュリティ教育は 受講したか の管理から、事故が起きにくくなる 運用へ変わる。

第V部

# 障害、インシデント、事業継続

有事の初動、連絡、復旧、継続を設計する。

第21章〜第23章

第21章

# 障害対応とインシデント管理

障害が起きた時、最初に失われやすいのはサーバーや回線だけではない。状況把握もすぐ失われる。誰かは調べている。誰かは利用者から問い合わせを受けている。誰かはベンダーへ連絡している。だが、全体としては 何が起きているか 誰が見ているか どこまで分かっているか が見えない。これが、ひとり情シスや小規模組織で最もつらい状態である。

第21章で扱うのは、この混乱を減らすための型である。深い攻撃対応は第22章で扱う。長時間停止や災害対応は第23章で扱う。第21章の中心は、その手前にある共通の incident management だ。障害かもしれないし、セキュリティ事故かもしれない。まだ原因は分からない。そういう時に、どう立ち上げ、どう連絡し、どう戻すかを整理する。

## 障害とセキュリティインシデントを切り分ける

第21章で最初に押さえたいのは、incident の意味である。ServiceNow は ITIL の incident を、IT service の `unplanned interruption` または `reduction in the quality` と説明している。Atlassian も incident を、service の品質低下や中断を引き起こし、emergency response を要する事象として整理している。つまり、incident とは、単なる問い合わせや不具合の別名ではない。通常運用を外れ、緊急対応として立ち上げるべき状態である。

ここで第16章との違いが出る。service request は、申請すれば進められる標準作業である。パスワード再設定、ソフト導入依頼、権限追加依頼はそちらに入る。incident は違う。メールが使えない、社内 Wi-Fi がつながらない、勤怠システムが遅い、SaaS　が一部落ちている。このように、業務へ影響が出ており、急いで状況整理と復旧が必要なものが incident である。

ただし、ここで難しいのが、障害とセキュリティインシデントの境界である。最初は単なる障害に見えても、実は不正アクセスや設定改ざんが原因かもしれない。CISA は Incident Response Playbook の対象を、confirmed malicious cyber activity だけでなく、悪性の可能性がまだ reasonably ruled out されていない状態も含めている。これは実務的である。原因未確定なら、早く ただの障害 と決めつけない方がよい。

第21章では、次のように考えると分かりやすい。

- 申請や定型作業なら request
- 業務影響を伴う停止や劣化なら incident
- 悪性の可能性が残るなら security incident 寄りで扱う

この切り分けがあるだけで、初動の迷いはかなり減る。

## 初動は、認知、起票、指揮、影響確認、初報の順で行う

障害対応で崩れやすいのは、原因究明から入ってしまうことである。もちろん原因は知りたい。だが、最初の 10 分から 30 分で先にやるべきことは別にある。Atlassian の incident response flow でも、detect の後に raise a new incident、open comms、assess、send initial comms を置いている。ここから分かるのは、原因究明より前に、incident を立ち上げて人を集める必要があるということだ。

初動では、次の順で考えるとよい。

1. 認知する
2. incident を起票する
3. 指揮役を決める
4. 影響範囲と代替手段を確認する
5. 初報を出す

起票はとても重要である。豪華な仕組みは要らないが、一つの incident ticket は必要だ。Atlassian も every incident should be tracked and documented としている。起票がないと、後から時系列がたどれず、誰がどこまで確認したかも分からなくなる。

指揮役も早く決めたい。ひとり情シスの組織でも、直す人 と 全体を見る人 を頭の中で分けるだけで違う。自分一人しかいないなら、自分がその二役を切り替えることになる。それでも、今は 調査モード なのか 社内告知モード なのかを意識して分けた方が混乱しにくい。

影響確認では、少なくとも次を早く押さえたい。

- 何が使えないか
- 誰が困っているか
- 全社か一部か
- 代替手段があるか
- セキュリティの疑いがあるか

この段階で完全な正解は要らない。仮説でよい。大切なのは、何も見えていないまま黙らないことである。

## 重大障害では、一つの source of truth を作る

重大障害で混乱が増える最大の理由は、情報が散ることだ。メール、チャット、口頭、電話、ベンダー窓口。どこにも部分情報はあるのに、全体像がない。これを防ぐには、一つの source of truth を持つのが最も効く。

Atlassian は、incident communication で dedicated status page を primary communication solution とし、clear source of truth を持つべきだとしている。これは status page 製品の話だけではない。ひとり情シスの現場では、incident ticket、共有チャット、更新メモのどれか一つを ここを見れば今の状況が分かる場所 と決めるだけでも効果がある。

重大障害では、少なくとも次を一か所に集めたい。

- 何が起きているか
- 影響範囲
- いまのステータス
- 次の更新予定
- 誰が見ているか
- ベンダー対応状況

連絡も早く出した方がよい。Atlassian Support は `communicate early` `communicate often` `stay consistent across channels` を勧めている。つまり、まだ調査中でも、起きていることと影響の仮説を短く伝え、次の更新時刻を示した方がよい。

初報は、たとえば次のくらいで十分である。

- 何の障害か

- 誰に影響があるか
- いま調査中か、回避策があるか
- 次回更新時刻

ここで避けたいのは、詳細が分かるまで黙ることだ。利用者は　まだ直っていないこと　より　何も分からないこと　に強い不安を感じる。初報は安心させるためというより、混乱を広げないために必要である。

## 復旧は、原因究明より先に業務影響を下げる

incident management の goal は、ServiceNow が説明する通り、normal service operation をできるだけ早く回復し business impact を最小化することである。つまり、最初の目的は美しい原因分析ではなく、業務影響を下げることだ。

ここで使う考え方は、次の三つである。

- workaround
- rollback
- restoration

workaround　は、完全には直っていなくても業務を回すための回避策である。別の回線を使う。別の申請手段へ切り替える。SaaS　障害なら手作業へ一時退避する。rollback は、直前変更が怪しい時に元へ戻す判断である。restoration は、サービスを intended state へ戻すことだ。

この順番が大切である。たとえば、社内メールが止まった時に、根本原因がまだ分からなくても、チャットや電話で代替連絡を案内すれば業務影響は下がる。SaaS　障害なら、ベンダー障害情報を確認しつつ、代替手段を先に出せる。原因は後で詰めればよい。

一方で、原因未確定のまま危険な変更を重ねるのは避けたい。特に、悪性の可能性が消えていない時はなおさらである。とりあえず再起動 とりあえず消す は、場合によっては状況を悪化させる。第22章へ渡す条件に触れそうなら、むやみに証跡を壊さない方がよい。

## 運用に戻すまでの判断を急がない

一度つながった、一度ログインできた、それだけで　close　しない方がよい。incident がつらいのは、復旧したと思ったら再発することだ。だから、運用に戻すまでにはもう一段階必要である。

ここで見たいのは、次の点である。

- 影響範囲は本当に収束したか
- 代替手段を戻してよいか
- 監視値や利用者報告は落ち着いたか
- 未解決課題は何か
- セキュリティ事故の可能性は消えたか

Atlassian も、incident is resolved を `affected service resumes functioning in its usual way` としている。つまり、単に一瞬戻っただけでなく、通常の状態へ戻ったかを見る必要がある。

また、close　の前には、何を保留して次の課題へ送るかを明確にした方がよい。恒久対策がまだ、監視改善がまだ、ベンダー回答待ち、レビュー未実施。こうしたものを曖昧にして close すると、後で何も残らない。

## 事後レビューは、責任追及でなく改善に使う

incident が終わると、そのまま次の仕事へ流れやすい。だが、振り返りをしないと、次も同じ混乱を繰り返す。Atlassian の postmortem は、impact、actions taken、root cause、follow-up actions を含む written record と整理している。さらに、責任追及を目的にしない postmortem を強調し、timeline、causal chain、mitigations を個人ではなく system と process で捉えるとしている。

ここは第21章の重要点である。振り返りは犯人探しではない。誰がミスしたかより、なぜそのミスが起きても止まらなかったか なぜ気づくのが遅れたか なぜ連絡が遅れたか を見る方が価値がある。

レビューでは、最低限次を残したい。

- 何が起きたか
- 誰にどんな影響があったか
- いつ何をしたか
- 根本原因と寄与要因は何か
- 何を直すか
- 誰がいつまでに直すか

ここで時系列があると強い。Atlassian の incident timeline の考え方が useful なのは、後から記憶で作文しなくてよいからである。障害中にメモしておけば、振り返りの質がかなり上がる。

また、振り返りの出力は改善へつながらなければ意味がない。監視追加、権限見直し、切り戻し手順改善、ベンダー連絡先整理、初報テンプレート修正。こうした改善項目を担当者と期限付きで残したい。

## 小さな会社で現実的に回るやり方

小さな会社では、NOC や専任の incident commander はいないことが多い。だが、それでも最低限の型は作れる。現実的には、次の五つで十分始められる。

- 共有チケットを一つ持つ
- incident 時に使うチャット部屋を一つ決める
- 初報テンプレートを一つ作る
- 更新テンプレートを一つ作る
- 振り返りメモの型を一つ持つ

これだけでも、かなり違う。ツールの名前は何でもよい。大切なのは、毎回ゼロから考えない ことだ。

severity も難しくしなくてよい。たとえば次の三段階で十分である。

- 高
  - 全社停止、顧客影響大、代替手段なし
- 中
  - 一部停止、代替手段あり、業務継続は可能
- 低
  - 限定影響、通常時間内対応でよい

このくらいでも、誰を呼ぶか、いつ初報を出すか、ベンダーへどれだけ急いで連絡するかの判断がしやすくなる。

## 通常時の例と、崩れた時の例

通常時の例では、社内メールが使えないという報告が来た時点で、incident ticket を起票する。影響範囲を 全社か、一部か で確認し、チャット部屋を一つ開き、担当と連絡役を分ける。初報では メール障害を調査中であること 影響範囲 代替連絡手段 次回更新時刻 を出す。ベンダー問い合わせも ticket に紐付け、時系列で残す。復旧後は数十分から数時間の monitoring を挟み、review でタイムライン、原因、改善項目を残す。結果として、直すだけでなく、周囲も状況を追える。

崩れた時の例では、利用者から メールが変だ と個別に連絡が来るが、誰も起票しない。ある人は再起動し、ある人はベンダーへ電話し、ある人は口頭で 直りそうらしい と言う。社内告知は出ず、営業は顧客へ返信できないまま待つ。途中で もしかして乗っ取りかもしれない という話も出るが、誰もその判断を引き取らない。夕方に一度使えるようになるが、原因も時系列も残っていない。翌週また似た障害が起きても、前回の教訓が使えない。問題は、技術力ではなく型がなかったことである。

## 最低限ここまではやる

第21章の内容を一度に全部整えなくてもよい。だが、次の五つは早めに持ちたい。

1. incident と request を分ける基準を持つ
2. 初動で起票し、指揮役を決める
3. 初報と更新のテンプレートを持つ
4. 重大障害では一つのタイムラインを残す
5. 振り返りで改善項目を担当者と期限付きで残す

この五つがあるだけで、障害対応はかなり安定する。障害対応とは、慌てて直すことではない。incident として立ち上げ、影響を把握し、連絡を出し、復旧し、記録を残し、次に同じ混乱を起こしにくくすることである。

## 今日、今週、後でやること

今日やることは、incident として立ち上げる条件と、初報で必ず書く項目を一枚にすることである。これがないと、毎回 これは大ごとか から迷う。

今週やることは、共有チケット、チャット部屋、更新メモの置き場を決め、障害時にそこへ情報を集約する練習をすることである。あわせて、security 可能性がある時に第22章寄りへ切り替える条件も決めたい。

後でやることは、振り返りの型を整え、実際の障害一件で短く回してみることである。ここまでできると、障害対応は属人的な火消しから、改善につながるインシデント管理へ変わる。

第22章

# ランサムウェア、情報漏えい、不正アクセス対応

重大セキュリティ事故では、最初にやるべきことが通常の障害対応と少し違う。障害なら、できるだけ早く戻したい。だが、ランサムウェア、情報漏えい、不正アクセスでは、急いで戻そうとした動きそのものが被害拡大や証跡消失につながることがある。再起動してしまう。感染端末を社内ネットワークへつないだままにする。乗っ取られたアカウントのパスワードだけを変えて安心する。漏えい範囲が固まるまで黙ってしまう。こうした初動ミスは、後から取り返しにくい。

第21章では incident management の共通型を扱った。第22章で扱うのは、その中でも 直す前に止める 消す前に残す 確定前でも報告判断を始める 必要がある事故である。第19章の法制度説明や第23章の BCP へ広げすぎず、ここでは重大セキュリティ事故の初動と連携に絞って整理する。

## ランサムウェアの被害像を甘く見ない

ランサムウェアという言葉を聞くと、ファイルが暗号化される攻撃 を想像しやすい。もちろんそれは正しい。だが、今の実務ではそれだけでは足りない。CISA の #StopRansomware Guide は、ransomware and data extortion としてガイドを整理している。JPCERT/CC も 侵入型ランサムウェア攻撃 という言い方で、攻撃者が内部ネットワークへ侵入した後に情報窃取や暗号化を行う型を区別している。つまり、ランサムウェアは 壊れた PC の修理ではなく、侵入、窃取、横展開、脅迫を含む事故 と見た方がよい。

この見方に変えると、最初に考えるべきことも変わる。暗号化された端末が一台見つかった時に、本当に一台だけなのか。共有フォルダはどうか。管理者アカウントは使われていないか。VPN や RDP はどうか。バックアップやクラウドストレージへ触られていないか。ここまで見る必要がある。

また、身代金要求も単純ではない。CISA と FBI の注意喚起は、支払いが復旧や情報非公開を保証しないことを明確にしている。したがって、払えば戻るかもしれない を前提に初動を組んではいけない。第22章で重要なのは、支払うかどうか以前に、被害拡大を止め、証跡を残し、経営判断に必要な材料を集めることである。

## 感染時の初動は、隔離、封じ込め、証跡保全の順で考える

ランサムウェアらしき症状が出た時、最初の目的は 直すこと ではない。まずは広がらないようにすることである。CISA の Ransomware Response Checklist は、影響を受けたシステムを直ちに隔離することを最初に置いている。複数システムや複数サブネットに広がっているなら、スイッチ単位でネットワークを落とす判断にも触れている。つまり、一台ずつ様子を見る　より、広がりを止める方が先である。

初動では、少なくとも次の順で考えたい。

1. 影響端末、共有先、接続元を隔離する
2. どこまで広がっているかを粗く把握する
3. 証跡を残す
4. 重要システムの復旧優先順位を決める
5. 外部支援を呼ぶ

隔離は、端末だけの話ではない。影響を受けたユーザーアカウント、共有フォルダ、VPN 接続、リモート管理経路、クラウド同期も視野に入れる必要がある。Wi-Fi を切る、LAN を抜く、対象 VLAN を切り離す。こうした動きは、復旧作業より先である。

ここで気をつけたいのが、電源断を乱発しないことだ。CISA　は、ネットワークから切り離せない時の最後の手段として power down に触れている。電源断は拡大防止に効くことがある一方で、揮発性メモリ上の証跡や実行中の痕跡を失うことがある。したがって、落とした方がよさそう　で反射的に切るのではなく、切り離し不能な時の手段として考えた方がよい。

証跡保全も早い方がよい。最低限でも、次は残したい。

- いつ気づいたか
- どの端末、どのアカウントで起きたか
- ransom note の内容
- 暗号化、削除、拡張子変更の痕跡
- 影響を受けた共有先やクラウド同期先
- 直前の不審なログイン、接続、権限変更

ここでは完璧なフォレンジックを目指さなくてよい。ひとり情シスなら、まず時刻、端末名、ユーザー名、画面表示、ログの場所を押さえるだけでも価値がある。JPCERT/CC や専門ベンダーへ相談する時も、何がいつ起きたかがあるだけで進めやすい。

反対に、避けたい行動もある。とりあえず再起動する。とりあえず初期化する。とりあえずバックアップをつなぐ。とりあえず共有フォルダへ再接続する。これらは、状況が分からないまま行うと悪化しやすい。特にバックアップは、汚染範囲を確かめる前に触ると、最後の復旧手段まで巻き込みかねない。

また、攻撃者が組織の対応を監視している可能性も考えたい。CISA は out-of-band communication を勧めている。重大なランサムウェアでは、普段使っているメールやチャットが見られているかもしれない。少なくとも初期の意思決定では、電話や別回線を使う前提を持った方がよい。

## 情報漏えい疑いでは、事実確認と報告判断を並行で進める

情報漏えい疑いの難しさは、発生直後に全部は分からないことだ。誤送信かもしれない。外部共有設定ミスかもしれない。乗っ取られたアカウントからダウンロードされたかもしれない。ランサムウェアで窃取された可能性もある。こういう時に 全部分かってから考える と遅れやすい。第22章では、事実確認と報告判断を並行で進める方がよい。

最初に整理したいのは、次の点である。

- 何のデータか
- 個人データに当たるか
- 何人分くらいか
- 閲覧、取得、外部送信が確認されたのか、それともおそれ段階か
- いまも外部からアクセス可能か
- 自社起因か、委託先起因か

ここでの 個人データ の整理自体は第19章で扱った。第22章で強調したいのは、事故時にはこの確認を急ぐ必要があるという点である。個人情報保護委員会は、個人の権利利益を害するおそれがある事態として、要配慮個人情報、財産的被害のおそれ、不正目的のおそれ、1,000人超を挙げている。そして速報は 速やかに（概ね3〜5日以内） としている。つまり、調査に一週間かけてから考える、という時間感覚ではない。

確報にも期限がある。個人情報保護委員会の整理では、通常は 30 日以内、不正の目的をもって行われたおそれがある場合は 60 日以内である。ランサムウェアや不正アクセスが絡むなら、この 不正の目的 の類型に入りうる。したがって、漏えい疑いの時点で、個人情報保護委員会への速報、確報、本人通知の要否を誰が判断するかを早く立ち上げた方がよい。

ここで誤解しやすいのが、公表である。個人情報保護委員会 FAQ は、公表が一律に義務付けられているわけではない一方、事案の内容に応じて望ましい場合があるとしている。また、公表が二次被害拡大につながる可能性があるなら、公表しないこともありうる。つまり、漏えいらしいから必ず公開 でも 公開義務はないから黙る でもなく、本人通知、二次被害、経営判断、広報判断を合わせて考える必要がある。

ランサムウェアでは、個人データの漏えい等又はそのおそれについて共通様式で報告できる仕組みもある。こうした制度面を知っているだけで、事故時の迷いは減る。

委託先起因の事故も同じである。SaaS 事業者、業務委託先、BPO 先で漏えい等が起きた場合でも、個人情報保護委員会 FAQ4-Q203 が示す通り、委託元の義務は消えない。ベンダーからの正式レポート待ちで止まるのではなく、自社として 何のデータが対象か 本人通知が必要か 社内で誰に連絡するか を先に動かす必要がある。

## 不正アクセス、乗っ取り、なりすましでは、アカウントだけでなく連携も止める

不正アクセスや乗っ取り対応で最も多い誤解は、パスワード変更で終わると思ってしまうことだ。実際には、それでは足りないことが多い。Microsoft Entra の緊急アクセス剥奪資料は、ユーザー無効化、refresh token の失効、端末無効化を並べている。つまり、止めるべきものは パスワード だけではなく、アカウント セッション トークン 端末 である。

まずやりたいのは、侵害された可能性のあるアカウントを止めることだ。必要に応じて一時無効化する。次に、既存セッションや refresh token を失効させる。さらに、そのユーザーが使っていた管理端末、メールクライアント、モバイル端末も見直す。管理者アカウントなら、権限棚卸しや緊急パスワード変更、MFA 再登録も要る。

OAuth 悪用も見落としやすい。Microsoft の consent phishing 資料では、悪意あるアプリを無効化しても、既存 access token は期限まで有効な場合があるとしている。つまり、怪しいアプリ同意を見つけて 削除したから終わり ではない。アプリ停止に加えて、ユーザー側のセッション失効やアカウント制御も必要になる。

Google Workspace でも同じ発想が必要である。Google は、特定アプリの block や、必要なら `Block all third-party API access` のような広い遮断ができると示している。緊急時には、業務影響と引き換えでも広めに止める判断が必要になることがある。

また、乗っ取りの痕跡は認証以外にも残る。最低限でも、次を確認したい。

- メール転送設定

- 受信ルール、仕分けルール
- 委任設定、共有設定
- 外部共有リンク
- OAuth 同意アプリ
- API トークンや連携設定

BEC もここに入る。取引先や経営者になりすましたメールで口座変更や送金を促す型は、単なる迷惑メールではない。IPA は BEC を独立した脅威として扱っているし、FBI は被害時に金融機関へ即連絡し、送金先金融機関への連絡を依頼するよう案内している。実務でもこれは重要である。送金が絡む時は、社内確認が終わってから では遅い。まず自社の金融機関へ連絡し、着金先への停止、組戻し、照会を急ぐ。その上で、警察、法務、経営へつなぐ方がよい。

## 法務、経営、広報、外部機関との連携を遅らせない

重大セキュリティ事故では、技術対応だけで閉じない。誰にどこまで連絡するかが、その後を大きく左右する。ここで遅れやすいのは、まだ確定していないから上げない という判断である。だが、重大事故では確定前でも初報が必要だ。

経営には、少なくとも次を早く渡したい。

- 何が起きているか
- どの業務に影響があるか
- 何がまだ不明か
- 次回更新時刻
- 外部連携が必要か

法務や個人情報保護の担当には、個人データ該当性、本人通知、公表、証跡保全、契約上の通知義務を早めに見てもらう。広報には、問い合わせが来た時の一次回答と、公表方針の整理が必要になる。委託先や SaaS 事業者には、ログ保全、セッション無効化、共有停止、追加調査を依頼する。第21章で述べた通り、一つのタイムライン、一つの source of truth はここでも重要である。

外部機関への連携も、抱え込まない方がよい。IPA は 各種インシデント発生時の初動対応に関する相談 を受け付けている。JPCERT/CC も、インシデント初動対応のサポートや被害箇所特定の支援を受け付けている。ひとり情シスが一人で判断し切れない時に、こうした窓口は現実に使える。

もちろん、全てを外へ出せばよいわけではない。法的な論点、対外公表、被害者連絡は、社内責任者と合わせて決める必要がある。ただし、全部自分で整理してから という順番にすると遅れやすい。重大事故では、社内外の関係者を早く立ち上げた方が、結局は速い。

## 小さな会社で現実的に回るやり方

小さな会社で、SOC やフォレンジックチームを前提にする必要はない。だが、重大事故用の最小セットは持った方がよい。現実的には、次の五つがあるだけでもかなり違う。

- 重大インシデント用の最初の 30 分チェックリスト
- 経営、法務、広報、ベンダー、金融機関の緊急連絡先
- 普段のメールやチャットが使えない時の連絡手段
- 個人情報保護委員会報告条件と期限の一枚メモ
- JPCERT/CC、IPA、外部専門家の相談先

これらは、大きな仕組みではない。だが、事故時に 何からやるか 誰へ上げるか が分かるだけで、初動の質は変わる。

また、復旧の優先順位も平時に決めておきたい。人事給与、会計、受発注、社内認証、メール、共有ファイル。どれから戻すかが決まっていないと、事故時に声の大きい要望へ引っ張られやすい。第23章の BCP ほど大げさでなくても、重大事故向けの簡単な優先順位表は持っておいた方がよい。

## 通常時の例と、崩れた時の例

通常時の例では、経理担当の PC に ransom note が表示された時点で、端末をネットワークから外す。共有フォルダとその担当アカウントの利用状況を確認し、同じ認証情報を使う経路も一時的に止める。incident ticket を立て、時刻、端末名、ユーザー名、画面表示、影響範囲を記録する。普段のチャットが見られている可能性も考え、別経路で関係者を集める。JPCERT/CC や専門ベンダーへ相談しつつ、バックアップや重要システムの汚染範囲を確認する。人事フォルダへのアクセス痕跡が見えた時点で、個人データ漏えい等のおそれとして社内の法務、個人情報保護担当、経営へ上げ、個人情報保護委員会への速報判断を始める。結果として、復旧より先に被害拡大防止と報告判断が立ち上がる。

別の通常時の例では、営業担当の Microsoft 365 アカウントから不審な送信が見つかる。まずアカウントを無効化し、refresh token を失効させ、利用端末も止める。同時に、メール転送、受信ルール、委任設定、OAuth 同意を確認する。怪しいアプリ同意が見つかったため、そのアプリを無効化し、影響ユーザーのセッションも切る。取引先へ偽請求書メールが出ていたため、営業部門、経理、法務へ連絡し、入金や送金に影響が出ていないかを確認する。こうすると、ログインを止めたら終わり で済まない実務が一本線で見える。

崩れた時の例では、経理 PC が暗号化されたが、とりあえず再起動する。社内 LAN にはつながったままで、ファイルサーバーにもアクセスし直す。バックアップ NAS も同じネットワーク上にあり、そのまま接続を続ける。社内への連絡は 一部 PC が不調です 程度で、セキュリティ事故として立ち上がらない。数時間後、共有フォルダと別部署端末にも被害が広がる。後から見ると、VPN アカウントと管理者権限の悪用もあったが、初動でログや画面記録が残っていない。人事データの流出可能性も出たが、誰も個人情報保護委員会報告の期限を把握しておらず、速報が遅れる。問題は、技術知識が足りなかったことより、重大事故の型がなかったことである。

## 最低限ここまではやる

第22章の内容を一度に全部整えなくてもよい。だが、次の五つは早めに持ちたい。

1. ランサムウェアや乗っ取り時に、まず隔離する対象と連絡手段を決める
2. 再起動や初期化の前に、時刻、端末、アカウント、画面、ログを残す
3. 個人データ漏えい等の報告条件と期限を一枚で見られるようにする
4. 乗っ取り時に、アカウント、セッション、端末、OAuth 連携を止める手順を確認する
5. 経営、法務、広報、ベンダー、金融機関、外部相談先の連絡網を持つ

この五つがあるだけで、重大セキュリティ事故の初動はかなり変わる。重大セキュリティ事故では、直すことより先に、被害拡大を止め、証跡を残し、報告判断を始め、社内外の連携を立ち上げる必要がある。

## 今日、今週、後でやること

今日やることは、ランサムウェア、乗っ取り、漏えい疑いの三つについて、最初の 30 分でやることを一枚にまとめることである。ここがないと、事故時に毎回ゼロから迷う。

今週やることは、個人情報保護委員会報告の条件と期限、金融機関や委託先を含む緊急連絡先、普段のチャットが使えない時の連絡手段を整理することである。あわせて、Microsoft 365 や Google Workspace で、セッション失効やアプリ遮断を実際にどこで行うか確認しておきたい。

後でやることは、ランサムウェア、OAuth 悪用、誤送信や外部共有ミスを題材にした机上演習を短く回すことである。ここまでできると、第22章の内容は知識ではなく、重大セキュリティ事故の初動として使える形になる。

第23章

# BCP、災害対応、復旧訓練

BCP　という言葉を聞くと、分厚い文書や様式を思い浮かべる人が多い。だが、ひとり情シスの現場で本当に必要なのは、立派な冊子ではない。地震でオフィスへ入れない。停電でサーバー室が止まる。回線が半日落ちる。取引先からの供給が止まる。感染症で出社人数が一気に減る。こういう時に、何を先に守り、どこまで止まってよくて、戻るまで何で回すかを決めておくことだ。これが BCP の本体である。

第21章では障害対応と incident management の共通型を扱い、第22章では重大セキュリティ事故の初動を扱った。第23章で扱うのは、その先で 長く止まりうる時に、どう持ちこたえて戻すか である。ここでは、重要業務、復旧優先順位、代替手段、訓練に絞って整理する。

## 重要業務と business impact を先に決める

BCP　が形骸化する最大の理由は、システム一覧から考え始めることだ。サーバー、NAS、SaaS、回線、端末。もちろんそれらは大事である。だが、本当に先に決めるべきなのは、何の業務をどれだけ止めると会社が痛いかである。

Ready.gov は、BIA を `time sensitive or critical business processes` が止まった時の影響を見積もり、recovery strategies に必要な情報を集めるものとして整理している。NIST の BIA 定義も同じで、業務機能と中断影響を分析するプロセスと見る。つまり、出発点は IT 資産ではなく、事業である。

ここで最初に整理したいのは、次のような問いである。

- 何の業務が止まると売上や受注に直撃するか
- 何の業務が止まると法令、契約、給与、税務で問題になるか
- 何の業務が止まると顧客対応が崩れるか
- 何の業務は翌日や翌週でも許容できるか

この問いに答えると、優先順位が見え始める。たとえば、受発注、顧客対応、会計締め、人事給与、勤怠、社内認証。会社によって違うが、全部が同じ重さではない。

内閣府の事業継続ガイドラインも、重要製品、重要業務を絞り込み、その復旧時間や復旧水準を考える流れを取っている。つまり、BCP とは 全部止めない計画ではなく、先に戻すものを決める計画 である。

また、影響は技術だけでは測れない。Ready.gov が挙げるように、損失は売上減少、追加費用、契約違反、顧客離れ、規制上の問題として現れる。ひとり情シスとしては、業務部門や経営と会話しながら、何時間止まると何が困るか を日本語で書けるようにした方がよい。

## RTO、RPO、RLO を業務から逆算して決める

重要業務が見えたら、次は復旧目標を決める。ここで出てくるのが RTO と RPO である。難しそうに見えるが、意味は単純だ。RTO は どれだけ止まれるか、RPO は どこまで巻き戻せるか である。内閣府ガイドラインでは、これに加えて どの水準まで戻すか を目標復旧レベルとして扱っている。

NIST の定義では、RTO は mission や business process に悪影響が出る前に recovery phase にいられる全体時間である。RPO は、outage 後にデータをどの時点まで戻す必要があるかである。これをひとり情シス向けに言い換えると、次のようになる。

- RTO
  - その業務は何時間、何日止まると痛すぎるか
- RPO
  - データは昨日まで戻ればよいのか、1時間前まで必要か
- 復旧水準
  - 完全復旧が必要か、まずは 50 パーセントの運転でもよいか

ここで大事なのは、目標を願望で置かないことだ。内閣府ガイドラインは、目標復旧時間と目標復旧レベルは、講じた対策により達成可能であり、対外的に説明できるものでなければならないとしている。つまり、1時間で全部戻す と書くだけでは意味がない。本当にその回線、バックアップ、代替端末、要員、ベンダー体制でできるのかを見なければならない。

また、RTO と RPO はシステム単位より業務単位で置く方が分かりやすい。たとえば、受注処理は 4 時間以内に再開したい、給与計算は翌営業日までによい、ファイル共有は当日中、会議室予約は翌週でもよい。この粒度で整理すると、後で IT 側の復旧順へつなげやすい。

## 現地復旧だけでなく、代替戦略を持つ

BCP で最も危ういのは、元の場所が戻れば何とかなる 前提で止まることである。だが、実際には戻らない時間がある。オフィスへ入れない。停電が長引く。主回線が落ちる。サーバー室へ入れない。だから、現地復旧だけでなく代替戦略を持つ必要がある。

内閣府ガイドラインは、現地復旧戦略だけでなく、代替拠点、OEM や提携先活用、ICT ツール、テレワークの活用を含む代替戦略を示している。つまり、BCP とは 元に戻す方法 だけでなく、戻るまで回す方法 を決めることでもある。

代替戦略は、大げさでなくてよい。小さな会社なら、たとえば次のようなものがある。

- オフィスへ入れない時はテレワークへ切り替える
- 主回線断の時はモバイル回線やテザリングで重要業務だけつなぐ
- 社内申請システムが止まった時は、一時的にスプレッドシートや紙で受ける
- 共有ファイルが止まった時は、重要フォルダだけ別手段で参照する
- 本社が止まった時は、別拠点か自宅で最低限の意思決定を行う

ここで重要なのは、何でも代替しようとしないことだ。全部を平常通りに回そうとすると失敗しやすい。受注だけは止めない、給与だけは遅らせない、対外連絡だけは維持する。このように、重要業務に絞って代替策を考える方が現実的である。

また、災害時は安全確保が前提である。内閣府ガイドラインも、業務継続や再開の前提として従業員や顧客の安全確保を置いている。したがって、BCP は 仕事を回せ の計画ではなく、安全を守りつつ仕事をどう絞って続けるか の計画である。

## IT DRP は BCP に従って作る

IT DRP は、IT 部門だけの復旧手順書のように見えやすい。だが、Ready.gov は、IT DRP を BCP と conjunction で作り、IT の recovery priorities と RTO は BIA から導くべきだとしている。つまり、IT DRP は BCP の下にぶら下がるべきもので、逆ではない。

この原則は、ひとり情シスにとってとても重要である。技術的に戻しやすいものから戻すと、業務優先順位とずれることがあるからだ。たとえば、ファイル共有は戻ったが、受注に必要な認証やネットワークが戻っていない。本番サーバーは起動したが、問い合わせ窓口が止まったまま。こうなると、復旧したようで業務は戻らない。

IT DRP を考える時は、少なくとも次を並べて見たい。

- どの業務を再開するためのシステムか
- そのシステムは何に依存しているか
- どの順で戻すと業務が再開できるか
- どの契約、認証、ライセンス、ベンダー支援が必要か

たとえば、受注再開には回線、認証、業務アプリ、データ、担当者の端末が要るかもしれない。給与再開には人事システム、認証、ファイル、プリンタ、振込データの出力が要るかもしれない。こうして業務から逆算すると、サーバーA NAS B という復旧順ではなく、受注を戻すにはこれが先 という順に変わる。

第15章で扱ったバックアップや復元手順は、ここで活きる。ただし第23章では、バックアップ方式の詳細より、それが業務再開に間に合うかどうかを見る方が重要である。

## 災害時の連絡、意思決定、対外説明を決めておく

長時間停止で詰まりやすいのは、技術より連絡と判断である。誰が発動を決めるのか。誰が全社連絡を出すのか。顧客には何をいつ伝えるのか。取引先や委託先にはどう知らせるのか。これが曖昧だと、技術復旧が進んでも会社は動きにくい。

Ready.gov は、emergency plans に communications planning、IT support and recovery、continuity plans を含めるべきとしている。つまり、BCP に連絡と情報発信は最初から入っていなければならない。

最低限でも、次は決めておきたい。

- 発動条件
  - どんな事象、どの影響で BCP を立ち上げるか
- 判断者
  - 誰が業務停止、代替運用、対外連絡を決めるか
- 緊急連絡網
  - 経営、管理職、現場責任者、ベンダー、主要取引先
- 代替連絡手段
  - 普段のチャットやメールが使えない時にどうするか
- 対外説明
  - 顧客や取引先へ何をどの単位で伝えるか

また、安否確認や出社可否も BCP の一部である。オフィスが無事でも、人が動けなければ業務は回らない。したがって、災害対応では システムが無事か だけでなく 誰がどこで働けるか を早く確認する必要がある。

ここでも一つの source of truth が効く。状況一覧、発動判断、代替手段、次回更新時刻を一か所に集めるだけで、混乱はかなり減る。

## 訓練、テスト、机上演習で計画を使えるものにする

BCP は、書いただけでは動かない。Ready.gov は、training, testing, exercises are essential components of preparedness としている。NIST の TT&E 定義も、要員が役割を理解し、計画の妥当性を演習し、システムの動作を確認する means と整理している。つまり、訓練はおまけではない。計画の一部である。

ここで訓練を三つに分けると分かりやすい。

- training
  - 誰が何をするかを覚える
- tabletop exercise
  - シナリオで判断や連絡を練習する
- test
  - 実際に回線切替、復旧、代替手段を試す

読み合わせだけでは弱点が見えにくい。Ready.gov の Testing & Exercises も、代替施設の設備が planned RTO に間に合うか、夜間にチームを呼び出せるか、実際に試さないと分からないと示している。これは本質的である。

小さな会社なら、次のような訓練から始めやすい。

- 緊急連絡網の呼び出しテスト
- 主回線断を想定したテザリングやモバイル回線切替
- オフィス入館不可を想定した在宅切替
- 共有ファイル停止時の手作業運用
- 年1回の机上演習

内閣府ガイドラインも、教育・訓練の結果として見つかった弱点や問題点を、是正・改善へつなげるべきとしている。つまり、訓練の目的は 実施した ことではない。弱点を見つけて修正することである。

## 小さな会社で現実的に回るやり方

小さな会社で、最初から全社 BCM を完成させるのは難しい。だからこそ、絞った方がよい。中小企業庁は、事業継続力強化計画を 中小企業のための取り組みやすいBCP と位置付けている。これが示しているのは、簡便な入り口から始めてよいということだ。

現実的には、まず次の四枚があるだけでもかなり違う。

- 重要業務と優先順位の表
- 業務ごとの RTO と代替手段の表
- 緊急連絡網
- 年間の訓練計画

重要業務は三つから五つでよい。RTO もまずは仮置きでよい。代替手段も、テレワーク、手作業、モバイル回線、別拠点のどれかで十分である。大切なのは、ゼロより一を作ることだ。

訓練も大がかりでなくてよい。中小企業庁が紹介している BCP 訓練企画シートのように、安否確認、初動対応、事業継続の三つに分けて机上演習を組むだけでも、実務には効く。

## 通常時の例と、崩れた時の例

通常時の例では、会社は重要業務を 受注 給与 顧客問い合わせ対応 の三つに絞っている。受注は 4 時間以内に再開、給与は翌営業日、問い合わせ対応は当日中という仮の RTO を置いている。主回線断ではモバイル回線へ切り替え、オフィス入館不可では在宅へ切り替え、共有ファイル停止時は一時的に限定フォルダと手作業で受ける。緊急連絡網は別経路でも届くようにし、年1回は 地震でオフィスに入れない シナリオで机上演習を行う。結果として、災害時に 全部止まった ではなく 何を先に回すか がすぐ決まる。

別の通常時の例では、停電でサーバー室が止まった時に、まず安全確認と出社可否を確認し、その上で給与と受注に必要な認証、回線、ファイル、アプリの復旧順を見直す。回線復旧が遅れる前提で、受注窓口だけはモバイル回線とノートPCで先に立ち上げる。給与処理は翌営業日へずらす判断を経営が承認し、顧客や取引先への説明は決めた文面で出す。これなら、完全復旧までの間も会社は持ちこたえられる。

崩れた時の例では、BCP はあるが、重要業務が書かれていない。停電でオフィスが止まると、誰も何を優先するか決められない。情シスはファイルサーバーから戻そうとし、営業はメールを先に求め、人事は給与が危ないと言い、経営は顧客向け説明を急ぐ。主回線断の代替手段も試したことがなく、テザリングの台数も不足している。連絡網は古く、夜間に誰ともつながらない。結果として、技術的に全部壊れていたわけではないのに、業務の再開は遅れる。問題は設備不足より、優先順位と訓練不足である。

## 最低限ここまではやる

第23章の内容を一度に全部整えなくてもよい。だが、次の五つは早めに持ちたい。

1. 重要業務を三つに絞る
2. その三つに RTO と代替手段を置く
3. BCP と IT DRP の優先順位をそろえる
4. 緊急連絡網と発動条件を一枚にする
5. 年1回の机上演習と、小さなテストを実施する

この五つがあるだけで、BCP はかなり実務になる。BCP とは、何を先に守り、どれだけの時間でどの水準まで戻し、戻るまでどう回すかを平時に決めて、訓練で使える形にしておくことである。

## 今日、今週、後でやること

今日やることは、止まると困る業務を三つだけ書き出し、それぞれに 何時間なら止まれるか と 戻るまで何で回すか を一言添えることである。これだけで BCP の骨格が見える。

今週やることは、その三つに必要なシステム、回線、担当者、代替手段をひも付け、緊急連絡網と発動条件を一枚へまとめることである。あわせて、主回線断やオフィス入館不可を題材にした短い机上演習の日時も決めたい。

後でやることは、代替回線、在宅切替、手作業運用、復旧順を実際に試し、結果を見直しへつなげることである。ここまでできると、第23章の内容は 読んだ知識 ではなく、長時間停止に耐えるための BCP になる。

第VI部

# 改善、自動化、AI、引き継ぎ

属人化を減らし、次の運営へ渡す。

第24章〜第27章

第24章

# 自動化、スクリプト、業務改善

ひとり情シスの仕事は、派手な大仕事より、小さな繰り返しで時間を削られやすい。入社のたびに同じアカウントを作る。毎月ライセンスを数える。週次で同じ管理画面を開いて異常を拾う。問い合わせ票を別の台帳へ転記する。どれも一回ごとは小さい。だが、積み重なると重い。そして手で回している限り、漏れやばらつきも起きやすい。

そこで自動化を考えたくなる。これは正しい。だが、第24章で強調したいのは、自動化は速くするためだけに行うものではないという点である。本当に重要なのは、同じやり方で回り、漏れが減り、誰が見ても追える形にすることである。ここを外すと、自動化は便利な黒箱になり、別の属人化を生む。

## 自動化すべき仕事の見つけ方

自動化が失敗しやすいのは、手当たり次第に始めるからである。まず見たいのは、その作業が本当に自動化向きかどうかだ。

判断しやすい観点は、次の五つである。

- 頻度が高いか
- 手順が毎回ほぼ同じか
- 入力と出力が明確か
- 例外が少ないか
- 漏れた時の痛みが大きいか

たとえば、入社時のアカウント発行、ライセンス棚卸し、週次レポート作成、定期リマインド送信、古い共有リンクの棚卸しは、自動化候補になりやすい。逆に、例外判断が多い作業、関係者ごとにやり方が違う作業、そもそも手順が決まっていない作業は、自動化前に標準化が必要である。

ここで大切なのは、決まっていない仕事を自動化しないことだ。決まっていない仕事を自動化すると、迷いがそのままフローやコードに埋め込まれる。すると後で例外が増え、結局人が横で補正する運用になる。自動化の前に、まず 誰が 何を見て どこで終わりとするか を決めたい。

実務では、次のように考えると選びやすい。

- まずは 月に何回もやっている単純作業 を選ぶ
- 次に 漏れると事故になる作業 を選ぶ
- その後で 人が判断すべき余地が少ない作業 を選ぶ

この順で選ぶと、効果が出やすい。

## スクリプト、API、ノーコードの使い分け

自動化といっても、手段は一つではない。Power Automate や Jira Automation のようなノーコード型もあれば、Apps Script や PowerShell のようなスクリプト型もある。GitHub Actions のように、コードと一緒にワークフローを管理する形もある。重要なのは、どれが優れているかではなく、どの仕事に向いているかである。

ノーコードやワークフロー型は、何かが起きたら、条件を見て、何かをする　仕事に向いている。Atlassian Automation も、rule は trigger、condition、action の三段でできていると説明している。つまり、起票、通知、承認、同期、定期実行のような流れには向いている。見える化しやすく、現場に説明もしやすい。

一方で、処理の分岐が増える、データ整形が多い、複数ページの集計が要る、再利用したい、Git　　で管理したい、という時はスクリプトの方が向く。Google Apps Script の best practices も、外部呼び出しの最小化や batching を前提に、処理の組み方で性能が大きく変わることを示している。つまり、自由度が高いぶん、設計責任も自分に返ってくる。

API　は、その中間ではなく、優先的に検討すべき正式な入口である。対象サービスに公式 API があるなら、まずそれを使う方がよい。画面の位置や文言に依存しにくく、再現性も高いからだ。ノーコードでも、裏では API やコネクタを使っていることが多い。したがって、選び方としては次の順が実務的である。

1. 公式 API や正式なコネクタがあるかを見る
2. それで足りるならノーコードかスクリプトかを選ぶ
3. それでも無理なら画面操作型を検討する

ここで見落としやすいのが、所有者である。Google Apps Script の installable trigger は作成者アカウントで動く。Power Automate でも所有モデルが安定性に影響し、重要または長時間動くフローでは　service　principal　を勧めている。GitHub Actions の scheduled workflow も、cron を最後に変えた user が notifications の actor になり、actor の状態が実行へ影響しうる。つまり、何で作るか と同じくらい 誰の権限で動くか が重要である。

## 手作業削減と事故防止を同時に考える

自動化で一番避けたいのは、速くなった代わりに危なくなることである。そのためには、ロジックだけでなく、設定、権限、失敗時の戻し方も一緒に考える必要がある。

最初にやりたいのは、設定値とロジックを分けることだ。Power Automate の environment variables は、hardcoding を避け、flows を portable and easier to maintain にするための仕組みとして説明されている。これは本質的である。URL、メールアドレス、接続先 ID、環境差分、API ベース URL のようなものは、コードやフローの中へ直書きしない方がよい。

秘密情報も同じである。GitHub Actions の secrets は reusable workflows へ自動で渡らないなどの制約があり、Power Platform でも secret 型の environment variable や DLP の考え方がある。ここから言えるのは、秘密情報は 何となく見えない場所 に置けばよいのではなく、明示的に管理しなければならないということだ。

また、自動化は 成功する時 だけでなく 失敗する時 を先に考えたい。最低限でも、次は持ちたい。

- dry run やテスト方法
- 途中失敗時にどう止めるか
- 二重実行しても壊れにくいか
- 手で戻す時は何を戻すか

たとえば、アカウント発行自動化なら、最初は 申請内容を整形して下書きまで にとどめる方法もある。いきなり作成、権限付与、端末登録まで全自動にする必要はない。半自動でも、漏れが減るなら十分価値がある。

さらに、接続の組み合わせにも注意がいる。Power Automate の DLP policy は、どの connector や desktop flow module を組み合わせられるかを管理し、違反 flow は suspend されうる。これは単なる管理機能ではない。自動化がデータ持ち出し経路になりうることを示している。第24章では、自動化を作る時点で、どの情報をどこへ渡してよいか を確認する必要があると書きたい。

## 運用改善を定着させる

自動化は、一回動けば終わりではない。むしろ問題はその後に出る。人が変わる。接続先が変わる。ライセンスが変わる。URL が変わる。通知先が死ぬ。第24章で重要なのは、これを前提に残すことだ。

Power Automate は co-owner を必要最小限にし、通常は run-only permissions の共有で足りる場面が多いとしている。Apps Script は shared drives を使えば group ownership に寄せられる。GitHub Actions は reusable workflows で重複を減らし、SHA pinning が stability and security の観点で safest option だとしている。これらを踏まえると、維持しやすい自動化には共通点がある。

- 担当者が明確
- 予備の管理者がいる
- 定義ファイルやコードの置き場所が決まっている
- 名前と説明がある
- 再利用部分が共通化されている

名前も重要である。`test2` `flow-final` `script_new` のような名前では、後で誰も分からない。何をする自動化か、いつ動くか、誰が見るかが分かる名前にした方がよい。

また、ルールやフローは更新後の確認も必要である。Atlassian Automation も、有効化した後に audit log の status を見て成功確認するよう案内している。つまり、自動化は 保存した時点 ではなく 一回流して確認した時点 で初めて運用に乗る。

## 自動化の証跡とメンテナンス

自動化が危ういのは、止まることそのものより、止まったことに気づかない時である。だから、証跡と監視が必要になる。

Power Automate では、run ごとの inputs と outputs を見ながら確認できる。一方で、run history は既定で 28 日保持である。Atlassian の audit log は 90 日保持で、それ以前は復元できない。Apps Script では execution log は開発向けで短く、multi-user production environment では Cloud Logging が preferred choice とされている。つまり、ログがあるか だけでなく どれだけ残るか が違う。

ここで最低限ほしいのは、次の四つである。

- 失敗時に通知が来る
- 実行結果を後から見返せる
- どの設定値で動いたか追える
- 止め方が分かる

さらに、量が増えるとクォータや制限も効く。Apps Script は quotas を超えると exception で止まりうる。Atlassian Automation も service limits を超えると throttled になる。Scheduled rules that can run at the same time の制限や、検索件数の上限もある。つまり、自動化は 一回通る だけでなく 継続的にその件数で回るか を見なければならない。

この意味で、メンテナンス対象として残したい情報は次のようになる。

- 担当者
- 通知先
- 実行頻度

- 依存する接続先
- 環境変数や secrets の場所
- 停止手順
- 手動代替手順

これがない自動化は、動いている時だけ便利で、止まった時に面倒を増やす。

## 小さな会社で現実的に回るやり方

小さな会社では、最初から複雑な自動化基盤を作る必要はない。むしろ、単純で、効果が見えやすく、止めても戻しやすいものから始めた方がよい。

最初の候補として向いているのは、たとえば次のようなものだ。

- 入社、異動、退職の依頼起票とチェックリスト生成
- ライセンス棚卸しの定期レポート
- 共有アドレスや問い合わせ箱からの転記
- 毎週の管理者向けレポート送信
- 契約更新や証明書期限の通知

ここで意図的に避けたいのは、いきなり基幹処理を全自動にすることである。まずは、通知、集計、棚卸し、下書き生成のような 壊れてもすぐ戻せる仕事 から始めた方がよい。その中で、担当者、通知、ログ、停止方法の型を作っていく方が、後から強い。

## 通常時の例と、崩れた時の例

通常時の例では、入社申請フォームが出た時に、自動で アカウント発行 貸与端末準備 初期権限確認 のチケットを起票する。申請内容から部署、入社日、上長を読み取り、標準的な権限候補を提示するが、最終付与は情シスが確認する。フローの担当者は個人ではなく共有管理アカウントか service principal に寄せ、通知先は情シス共有アドレスにする。接続先 URL や対象グループは environment variable で分け、変更時にロジックを触らない。こうすると、全自動でなくても漏れは大きく減る。

別の通常時の例では、毎週 1 回、スクリプトが Microsoft 365 や Google Workspace からライセンス割り当て情報を取得し、休職者や退職済み候補、未使用ライセンス候補を一覧化する。結果は表に出して共有し、除外対象は別リストで持つ。スクリプト本体は共有リポジトリへ置き、接続情報は secrets で持つ。実行ログと失敗通知も残す。これなら、棚卸しは人が 見る 仕事になり、毎回集める 仕事ではなくなる。

崩れた時の例では、ある社員が自分の Google アカウントで Apps Script を作り、毎朝自動でレポートを送っていた。動いている間は便利だったが、異動後に権限が変わり、installable trigger はその人の権限で動いていたため止まる。通知先も本人だけで、誰も気づかない。別のフローでは、API キーと URL が直書きされており、環境移行時に全部を手で直す。さらに、run history は既定期間を過ぎて消え、原因も追えない。問題は自動化したことではなく、所有者、設定値、証跡を設計せずに作ったことである。

## 最低限ここまではやる

第24章の内容を一度に全部整えなくてもよい。だが、次の五つは早めに持ちたい。

1. 自動化候補を頻度と例外率で選ぶ
2. 重要な自動化を個人アカウント一つに依存させない
3. URL や接続先や秘密情報を直書きしない
4. 失敗通知と実行履歴の確認場所を決める
5. 止め方と手動代替手順を一行でも残す

この五つがあるだけで、自動化はかなり実務的になる。自動化とは、手作業を減らすことではない。標準化された仕事を、壊れにくく、追跡できる形で回すことである。

## 今日、今週、後でやること

今日やることは、毎週または毎月繰り返している作業を五つ書き出し、頻度 例外率 漏れた時の痛さ の三点で見直すことである。ここで一つだけ、自動化候補を選ぶ。

今週やることは、その候補について、担当者、通知先、停止手順、手動代替手順、設定値の置き場所を決めることである。あわせて、ノーコードで足りるのか、スクリプトが必要か、公式 API があるかも確認したい。

後でやることは、自動化した処理を　reusable　な形へ寄せ、ログ保持期間やクォータも見直しながら、二つ目、三つ目の改善へ広げることである。ここまでできると、第24章の自動化は 便利な個人技 ではなく、運用改善の基盤になる。

第25章

# 可視化、指標、コスト最適化

ひとり情シスの仕事は、実際にはかなり多くのものを扱っている。問い合わせ対応、端末配布、権限付与、退職処理、ライセンス棚卸し、契約更新、請求確認、障害対応、ベンダーとのやり取り。だが、これらはしばしば別々の場所に散っている。チケットには問い合わせ件数がある。管理画面にはアカウントやライセンスの状況がある。請求書には金額がある。更新日はカレンダーやメールの奥に埋もれる。こうなると、忙しいことは分かっても、何に時間が吸われ、何が改善され、何にムダがあるのかは見えにくい。

第25章で強調したいのは、可視化は報告資料を作るための仕事ではないという点である。本当に重要なのは、何を続け、何を止め、どこへ時間と予算を振るかを決められる状態を作ることである。数字がないと、改善も説明も属人的になる。逆に、数字がありすぎても判断は鈍る。したがって第25章では、全部を見える化する　ではなく、意思決定に必要なものだけを継続的に見える化する　ことを軸にしたい。

## 情シス業務を見える化する

見える化で最初にやるべきことは、グラフを作ることではない。何を一件として数えるか、どの単位で持つかを決めることである。

たとえば、次のものは分けて持った方がよい。

- 依頼
- インシデント

- 変更
- 資産棚卸し
- 契約更新
- 請求確認

この区別がないと、数字が混ざる。新規アカウント発行と障害復旧を同じ 問い合わせ件数 に入れると、件数は出ても中身が分からない。ドメイン更新や回線更新のような期限管理も、依頼件数の中に埋めると見えなくなる。したがって、まずは どの種類の仕事か を分けたい。

小さな会社で最初に持つべき見える化単位は、五つで十分である。

- 人
- サービス
- 契約
- コスト
- 期限

人では、誰が使っているか、休眠か、退職済みかを見る。サービスでは、どのツールや回線やシステムがあるかを見る。契約では、年契約か月契約か、解約期限はいつかを見る。コストでは、月額か従量か、誰の意思決定で発生しているかを見る。期限では、更新日、請求日、証明書期限、ドメイン期限、保守期限を持つ。

ここで重要なのは、可視化のために新しい巨大な仕組みを作らないことである。第25章の段階では、最初から BI 基盤を導入する必要はない。ひとり情シスに必要なのは、次が一つの流れで見えることだ。

- 今月どんな仕事が何件あったか
- どこで滞留しているか

- 何にいくら払っているか
- 使っていないものは何か
- 何がいつ更新されるか

これが見えれば、すでに十分に実務的である。

## 何を KPI として追うか

数字を集め始めると、次に起きやすい失敗は、全部を KPI と呼んでしまうことである。だが、KPI と 運用指標 は分けた方がよい。

KPI は、少数でよい。月次や四半期で、方向性が良いか悪いかを判断するものだ。一方、運用指標は、日々の詰まりや原因を見るためのものだ。件数、SLA、再オープン率、CSAT、自己解決率、未使用ライセンス率、クラウド予算差異など、どれも重要だが、全部を同じ重さで扱うと焦点がぼける。

見やすく整理すると、指標は次の六種類に分けられる。

- 量
- 時間
- 品質
- 自己解決
- リスク
- コスト

量には、問い合わせ件数、変更件数、未処理件数、滞留件数の増減がある。時間には、初回応答までの時間、解決までの時間、SLA 達成率がある。品質には、再オープン率、CSAT、初回解決率がある。自己解決には、ナレッジ閲覧や自己

解決できた件数のような指標がある。リスクには、権限棚卸し完了率、バックアップ復元テスト実施率、期限切れゼロかどうかがある。コストには、ライセンス使用率、1席あたりコスト、サービス別月額、予算差異、異常支出がある。

Atlassian の一次情報でも、Jira Service Management の標準レポートは Workload、Satisfaction、Requests deflected、Requests resolved を持っている。またダッシュボード指標として、SLA 達成率、SLA breach 率、first contact resolution、reopened rate、backlog growth まで定義している。これは示唆的である。つまり、たくさん来た だけでは足りず、間に合ったか やり直しが起きたか 自己解決できたか を一緒に見るべきなのである。

小さな会社で最初に置く KPI は、たとえば次のようなもので十分である。

1. 月次の滞留件数が増えていないか
2. 主要な依頼の SLA を守れているか
3. 再オープン率や手戻りが下がっているか
4. 未使用ライセンスや休眠アカウントが減っているか
5. 更新漏れや予算超過の予兆を早く拾えているか

そしてその下に、補助指標を置く。たとえば 問い合わせ件数を部署別に見る どの依頼種別が増えたか見る どのツールで休眠率が高いか見る といった具合である。

ここで大事なのは、指標の定義を曖昧にしないことだ。未使用ライセンス を、30 日無活動とするのか、90 日無活動とするのか。解決時間 を、受付から完了までとするのか、営業時間だけで測るのか。自己解決 を、記事閲覧数で見るのか、deflection で見るのか。定義が曖昧だと、前月比較も説明も崩れる。

さらに、レポートの鮮度にも注意がいる。Microsoft 365 の usage report は通常 24〜72 時間の遅れがあり、Google Workspace の reports も 1〜3 日、指標によっては 1〜6 日遅れる。つまり、これらはリアルタイム監視には向かない。第25章では、即時対応の数字 と 週次・月次棚卸しの数字 を分けて考えるべきだと書きたい。

## ライセンス、回線、クラウド費用の見直し

コスト最適化というと、安い製品へ乗り換える話だと思われがちである。だが、ひとり情シスの実務で先に効くのは、使っていないもの、契約条件と合っていないもの、所有者が曖昧なものを整理することである。

最初に見たいのは、契約している数 と 実際に使っている数 が一致しているかである。Microsoft 365 の activity reports は、サービスをほとんど使っていないユーザーを把握し、ライセンス見直しの候補を拾う材料になる。Google Workspace でも、Users Usage Report や各種 usage data から last login や利用実態を長めの期間で見られる。したがって、次のような棚卸しが可能になる。

- active だが実利用が少ないユーザー
- suspended や archived へ寄せるべきユーザー
- 退職済みだが add-on ライセンスが残っているユーザー
- guest や委託先で不要になった利用者

ただし、利用実態だけ見て席数を減らせばよいとは限らない。ここで契約条件が効く。Google Workspace の一次情報では、Flexible Plan は commitment なしだが、Annual/Fixed-Term Plan は 1 年以上の commitment があり、更新時まで minimum number of user licenses を減らせない。つまり、未使用アカウントを消したこと自体には意味があるが、年間契約なら今月の請求がすぐ下がるとは限らない。この違いを知らないと、減らしたのに安くならない という誤解が起きる。

退職者対応でも同じである。Google Workspace の archived user は active license を 24 時間以内に再利用可能にしつつ、データを保持できる。一方で、archived user のデータは pooled storage に残り続ける。したがって、第25章では席を空ける と 保存容量を減らす は別問題だと書いておきたい。

固定費の見直し対象は、SaaS だけではない。回線、VPN、MDM、EDR、バックアップ、ドメイン、DNS、証明書監視、保守契約、SMS 認証、クラウド commit なども並べて見た方がよい。ここでは 高いか安いか より、次を見たい。

- いま誰が使っているか
- なくすと何が止まるか
- 代替があるか
- 契約条件は何か
- 更新判断をいつする必要があるか

クラウド費用はさらに、請求を見た後に知る から抜ける必要がある。AWS は cost anomaly detection で anomalous spend を検知し、service、account、region、usage type ごとに根本原因候補を見られる。Google Cloud budgets は actual cost と planned cost を比べ、alert threshold や Pub/Sub 通知を設定できる。Azure も exports や query API を前提に、タグ、management groups、custom dimensions で cost allocation することを勧めている。

ここで強く書いておきたいのは、予算アラートは自動停止ではないという点である。Google Cloud も、budget は spend を自動で cap しないと明記している。AWS anomaly detection も、検知まで最大 24 時間程度の遅延がありうる。したがって、予算や異常検知は 安心の仕組み ではなく 動くきっかけ である。通知を誰が受け、どこを見て、どこまで掘るかを決めなければ効かない。

クラウド費用では、総額だけでなく、誰の責任範囲かが分かるようにしたい。AWS は cost allocation tags や cost categories、Azure は tags や management groups、Google Cloud は project や labels で scope を切れる。第25章では、タグがないと分析できない という言い方より、責任の所在が分からない費用は削減も説明もできない と書く方が伝わる。

## 契約更新と棚卸しを定期化する

コスト最適化が失敗しやすいのは、更新の直前に慌てて判断するからである。ひとり情シスが持つべきなのは、単なる契約一覧ではなく、更新判断に必要な情報が入った更新カレンダーである。

最低限、次は持ちたい。

- 契約名
- サービス名
- 担当者
- ベンダー
- 月額または年額
- 契約形態
- 更新日
- 解約期限
- 最低利用期間
- 自動更新の有無
- 継続、縮小、停止の判断期限

ここでのコツは、更新日だけでなく、いつまでにやめる判断をしなければならないか を持つことである。たとえば年契約の SaaS は、更新日の 30 日前や 60 日前に解約意思表示が必要なことがある。回線や保守契約は、切替や撤去に時間がかかる。したがって、見るべき日は一つではない。

実務では、次の四段階に分けると回しやすい。

- 90 日前に利用実態と代替可否を見る
- 60 日前に継続、縮小、停止の仮判断を置く
- 30 日前に決裁とベンダー連絡を済ませる
- 7 日前に請求、設定、通知先の最終確認をする

更新判断の材料も、金額だけでは弱い。最低でも次が必要である。

- 現在の利用者数
- 実際の利用頻度
- 代替手段の有無
- 停止した時の影響
- 契約条件
- データ引き上げや移行の難易度

これがあれば、何となく継続　を減らせる。逆に、これがない更新は、使っていないのに続くか、本当は必要なのに切って事故になるかのどちらかになりやすい。

ドメイン、証明書、DNS、回線、SaaS、セキュリティ製品、クラウド　commit は、部署横断で影響が出ることが多い。第25章では、更新カレンダーを ひとり情シスだけの　TODO　ではなく、会社の継続運営に必要な管理表として位置づけたい。

## 経営へ説明できる数字を持つ

情シスの仕事は、数字がないと　いろいろやっているらしい　で終わりやすい。だが、件数だけの報告でも弱い。問い合わせ件数が 100 件だったとしても、それだけでは多いのか少ないのか、改善したのか悪化したのか、経営には伝わりにくい。

説明に必要なのは、量 時間 品質 費用 価値 をつなぐことである。たとえば次のように言い換える。

- 問い合わせは 100 件だった
- そのうち 35 件はパスワードや入社手続きで、自己解決導線を作れる類型だった
- backlog growth は前月より下がった
- SLA 達成率は維持した
- 再オープン率は下がった
- その結果、月末の滞留が減り、翌月の負荷平準化につながった

これなら、単なる件数報告ではなくなる。

コストも同じである。年額 30 万円下げた だけでなく、退職者と休眠ユーザーを整理して未使用席を解消した 年間契約なので即減額ではないが、次回更新時の縮小余地を確定した 今後 12 か月の固定費見込みを下げた とつなぐ方が、意思決定に耐える。

ここで役立つのが、unit economics 的な見方である。これは難しい話ではない。小さな会社なら、たとえば次のような見方で十分である。

- cost per active seat

- cost per request resolved
- cost per endpoint managed
- cost per new employee onboarded
- cost per branch office maintained

重要なのは、比較のために無理に全部を同じ単位へ押し込まないことだ。FinOps の一次情報でも、異なる事業やサービスを無理に横並び比較するより、同じ対象を時間推移で見る方が実務的だとしている。第25章でも、前月比 四半期比 同じ対象の改善前後 を重視した方がよい。

また、経営向けの数字は少なくてよい。現場向けには詳細が必要だが、役員向けに毎回 20 指標を出す必要はない。むしろ次のような一枚の方が伝わる。

- 今月の主要運用量
- 今月の主要品質
- 今月の主要コスト差異
- 今後 90 日の更新とリスク
- 先月からの改善と次の打ち手

これは chargeback のように厳密な部門課金をする話ではない。小さな会社では、まず `showback`、つまり どこで何が使われているかを見せる だけでも十分に意味がある。

## 通常時の例と、崩れた時の例

通常時の例では、月次レビュー用に一枚の運用サマリーを持つ。左側には依頼、インシデント、変更の件数と前月差、滞留件数の増減、SLA 達成率、再オープン率を置く。右側には主要 SaaS の契約数、active 利用者数、休眠候補、来月以降

90　日の更新予定を置く。下段にはクラウド費用の予算差異と異常検知の有無を置く。数字は多く見えるが、役割ごとにまとまっているため、何が増えたか　どこが詰まっているか 何を見直すか がその場で決まる。

別の通常時の例では、Microsoft 365 と Google Workspace の usage report を使い、90 日以上実利用が少ない候補を毎月洗い出す。退職者は archive 候補へ、休職者は停止候補へ、委託先終了者は削除候補へ分ける。ここで契約条件も横に置き、Flexible plan なら翌月反映を狙い、Annual/Fixed-Term なら次回更新での縮小候補として積み上げる。これなら、今月すぐ下がる費用 と 次回更新で下がる費用 を混同しない。

クラウド費用の通常時の例では、Google Cloud budgets や AWS anomaly detection　　の通知を情シス共有アドレスと経理担当へ送る。通知が来たら、project、service、region、usage type、invoice reconciliation を見て原因を切り分ける。月末には cost table や export を使って、どのプロジェクトや SKU が増えたかを見る。これなら、何か高い から どこが原因か まで進める。

崩れた時の例では、問い合わせ件数だけを毎月報告していたため、件数は横ばいなのに滞留件数が積み上がっていることに誰も気づかない。再オープン率も見ていないので、同じトラブルが繰り返し開き直されている。別の会社では、年間契約の SaaS で unused seat を削除したので来月から請求が下がると思っていたが、実際には更新時まで減額されず、経営への説明で詰まる。どちらも問題は数字不足ではない。数字の定義と見方が足りなかったのである。

さらに別の失敗では、Google Workspace や Microsoft 365 の usage report を即時監視のつもりで見ており、レポート遅延の存在を考えていなかった。そのため、当日の異常を　レポートに出ていないから問題なし　と誤解する。第25章では、これは避けたい。可視化は万能ではなく、鮮度と保持期間の前提がある。

## 最低限ここまではやる

第25章の内容を一度に全部整えなくてもよい。だが、次の五つは早めに持ちたい。

1. 依頼、インシデント、変更、更新、請求を分けて数える
2. KPI を少数に絞り、定義を一行でも書く
3. ライセンス利用実態と契約条件を同時に見る
4. 予算アラートや異常通知の受け先と初動を見る人を決める
5. 更新カレンダーに解約期限と担当者を入れる

この五つがあるだけで、情シス運用はかなり見えやすくなる。可視化とは数字を並べることではない。次の判断ができる状態を、継続的に作ることである。

## 今日、今週、後でやること

今日やることは、いま毎月見ている数字を十個書き出し、そのうち　意思決定に使っている数字　と　惰性で見ている数字　を分けることである。ここで、残す指標を五つに絞りたい。

今週やることは、主要 SaaS を一つ選び、契約席数、active 利用者数、休眠候補、契約形態、更新日、解約期限を一枚で見えるようにすることである。あわせて、問い合わせ系では滞留件数、SLA、再オープン率のどれか一つを新たに見える化したい。

後でやることは、クラウド予算通知や usage report を自動取り込みし、月次レビュー資料を半自動化しながら、総額 だけでなく 1席あたり 1件あたり の見方へ広げることである。ここまでできると、第25章の可視化は 数字を集める作業 ではなく、改善と説明の基盤になる。

第26章

# 生成 AI 活用と AI ガバナンス

生成 AI は、導入のハードルが低い。ブラウザを開けばすぐ使える。文章の下書き、要約、翻訳、検索補助、会議メモ、社内文書の問い合わせ対応まで、効果も見えやすい。だからこそ、企業では 気づいたら各自が使っている 状態になりやすい。だが、ここで見落としやすいのは、生成　AI　が単なる便利なチャットではないという点である。生成 AI は、情報を入力し、処理し、生成物として外へ出す、新しい情報経路である。

第26章で強調したいのは、生成 AI の論点は 使うかどうか より前に どの条件なら使ってよいか を決めることだという点である。プロンプトの書き方より先に、承認済みサービス、入力できる情報、共有範囲、外部接続、ログ、保持、監査を決めた方がよい。ここを曖昧にすると、便利さの裏で、個人情報の入力、機密の混入、過剰共有の可視化、外部 API への送信、出力の誤用が同時に進む。

## 生成 AI を社内導入する前に決めること

生成　AI　を導入する前に、最初に決めるべきものは　どのツールを入れるか　ではない。何のために使うか 誰が使うか 何を入れてよいか どこまでつながるか である。

用途は大きく分けると、次の四つで考えると整理しやすい。

- 一般的な文章作成や要約の補助
- 社内文書や社内データを参照する検索補助
- 外部サービスとつながる workflow や agent
- 顧客向け、公開向けの生成

この四つは、同じ 生成 AI でもリスクが違う。一般的な文章補助は比較的始めやすい。だが、社内文書検索になると権限と過剰共有の問題が入る。外部サービスとつながる actions や apps、connectors になると、第三者送信や誤操作の問題が入る。顧客向けや公開向けになると、誤情報、権利侵害、ブランド毀損の問題が強くなる。したがって、まずは用途で分けたい。

次に、承認済みサービスを定める。ここで重要なのは、生成 AI を一括りにしないことだ。OpenAI の個人向け ChatGPT、ChatGPT Business、ChatGPT Enterprise は同じではない。Google でも Gemini in Workspace、Gemini app、NotebookLM、Gem sharing は同じではない。Microsoft でも Copilot Chat、本体機能、connectors、custom agents は同じではない。学習利用、保持期間、監査、共有、管理者設定が違うからである。

そのため、導入前に最低でも次は確認したい。

- 入力データが学習に使われるか
- prompts と responses の保持期間はどうなるか
- 保存先や適用法令に注意点はあるか
- ログや監査証跡を取得できるか
- 共有範囲を制御できるか
- 外部 app、action、connector を制限できるか
- 退職や異動時に所有者や設定を回収できるか

総務省・経済産業省の AI 事業者ガイドライン 第1.1版は、AI 活用においてリスクの大きさに応じた対策を取る リスクベースアプローチ を示している。ここで言いたいのは、全部禁止 か 全部自由 かの二択ではなく、用途とリスクに応じて線を引くべきだということである。ひとり情シスに必要なのも、この線引きである。

また、契約確認も後回しにしない方がよい。経済産業省の AIの利用・開発に関する契約チェックリスト は、データの利用範囲、AI 生成物の利用条件、ログ保存、監査、セキュリティ、規約改定まで確認するよう整理している。つまり、生成 AI の導入は SaaS を一つ増やす ではなく、新しい情報処理先と契約する ことである。

## 入力禁止情報と利用ルール

次に必要なのは、入力禁止情報と利用ルールである。ここでありがちな失敗は、禁止事項だけを書いて終わることだ。だが実際には、何を入れてはいけないか と同じくらい どの条件なら使ってよいか を書いた方が運用しやすい。

入力禁止情報として早めに分けたいのは、たとえば次のようなものである。

- 個人情報
- 要配慮個人情報
- 顧客との契約情報
- 未公開の経営情報
- 営業秘密
- ソースコードや設計情報
- 認証情報、API キー、秘密情報
- 第三者から預かった非公開データ

個人情報保護委員会の注意喚起は、個人情報取扱事業者が個人情報を含むプロンプトを入力する時、利用目的達成に必要な範囲内かを確認すること、本人同意なく個人データを入力し、応答結果以外の目的で取り扱われる場合は、法令違反

となる可能性があることを示している。したがって、第26章では　個人情報を入れるな　とだけ書くのでは足りない。利用目的、同意、機械学習への利用有無、出力の不正確さまで含めて判断する必要がある。

ここで実務的なのは、禁止 条件付き許可 許可 の三段で持つことだ。

- 禁止
  - 顧客個人情報、要配慮個人情報、秘密情報、認証情報
- 条件付き許可
  - 社内資料の要約、匿名化した事例、公開前提でないが秘匿度の低い文案
- 許可
  - 公開済み資料の要約、一般的な文面の下書き、公開情報の整理

この三段にすると、現場は迷いにくい。

また、入力だけでなく出力の扱いも決めたい。生成　AI　の出力は、そのまま正しいとは限らない。個人情報保護委員会も、生成　AI　の出力には不正確な内容の個人情報が含まれるリスクがあると注意している。したがって、次のような用途では必ず人手確認を入れた方がよい。

- 顧客へ送る文面
- 契約、規程、法務文書
- 対外公開する説明文
- 評価、査定、懲戒に関わる文面
- セキュリティ設定やコード変更提案

著作権や出典確認も同じである。生成　AI　はもっともらしい文章を作るが、出典や権利関係が自動で正しく整理されるとは限らない。第26章では、出力は下書きであり、確定文ではない という姿勢を強く置いた方がよい。

## 権限、ログ、監査、過剰共有の管理

生成 AI の大きな論点は、入力情報だけではない。権限、共有、ログ、監査も同じくらい重要である。

ここで見落としやすいのが、生成 AI は新しい権限を作るより、既存の広すぎる権限を目立たせることが多いという点である。Microsoft は Microsoft 365 Copilot の oversharing を、導入時の代表的なリスクとして扱っている。理由は単純で、Copilot は既にアクセスできる情報を横断的に拾いやすくするからである。つまり、SharePoint や OneDrive の共有が広すぎれば、今まで 探さないと見つからなかった情報 が、AI によって すぐ見つかる情報 になる。

したがって、社内文書検索型の AI を入れる前には、次を見直したい。

- 誰でも見えるフォルダや共有リンク
- 全社共有に置かれた個人資料
- 退職者の所有物や放置共有
- 管理部門だけの資料が広く見える状態
- guest や外部共有が残っている場所

Microsoft Copilot connectors の資料も、permission setting が intended visibility model と合っているかを確認する必要があり、`Visible to everyone` は oversharing につながると明示している。つまり、AI を入れる前の本当の準備は、プロンプト研修より ACL の掃除である。

Google でも同じである。Gem sharing は Google Drive の共有設定に乗る。Google は Shared Gems are stored and shared in Google Drive と明記しており、外部共有を許していれば Gems も外へ共有されうる。したがって、カスタム AI の共有は、その機能単体ではなく、Drive や既存共有設定と一体で管理する必要がある。

OpenAI の GPTs でも、workspace owners は GPT sharing level、third-party GPTs、GPT actions の domain を制御できる。ここで重要なのは、社内 GPTと 外部 GPTを分けることだ。外部 GPT は custom actions や web browsing を持ちうるため、社内の GPT 設定だけでは統制できない部分がある。第26章では、次を分けて管理したい。

- 本体チャット
- 社内用カスタム GPT / Gem
- 外部提供の GPT
- actions / apps / connectors
- 社内文書検索や grounded search

ログと監査も、サービスごとの差が大きい。Microsoft 365 Copilot Chat は auditing / eDiscovery のために interaction を記録するとしている。Google Workspace は Gemini の audit logs を提供している。OpenAI も Business / Enterprise で workspace settings、usage analytics、Compliance API、custom GPT の所有者管理を持つ。だが、保持期間や取得できる粒度は同じではない。したがって、第26章では 使えるか の前に 誰が何をしたか後で追えるか を確認すべきだと書きたい。

さらに、カスタム GPT や Gem は 個人の便利設定 として放置しない方がよい。OpenAI の shared edit access は、共同管理、ownership reassignment、unowned GPTs　の把握を前提にしている。これは示唆的である。つまり、業務で使うカスタム AI は、台帳に載せ、所有者、編集者、共有範囲、外部接続、停止方法を持つべきなのである。

## AI の利用をめぐるサイバーリスク

生成 AI のセキュリティは、入力情報が漏れる だけではない。出力、接続、外部コンテンツの取込みでも崩れる。

まず避けたいのは、hallucination を事実として扱うことだ。生成 AI は自然な文章を出すが、その文章の正しさまでは保証しない。特に、FAQ　回答、規程解釈、契約要約、障害原因分析、設定提案のような用途では、人手確認が必須である。

次に、prompt injection を意識したい。これは、メール、添付、Web ページ、共有文書、外部サービスから取り込んだ内容の中に、AI の挙動を変える指示や誘導が埋め込まれる問題である。第26章では、ここを難しい研究話としてではなく、信頼していない外部コンテンツを AI に読ませる時の注意として書く方が実務的である。特に次は危ない。

- 外部から届いたメール本文の要約
- 添付ファイルの自動処理
- Web 検索付きの要約
- 外部アプリと連携する action
- 社内外が混ざる shared knowledge

さらに、actions や apps は便利だが、同時に第三者送信の入口でもある。OpenAI も、GPT が external APIs や apps を使う場合、入力の一部が third-party service へ送られることがあると明示している。したがって、本体チャットは承認と 外部 app 連携も承認 は同義ではない。ここを分けないと、業務データが思わぬ外部サービスへ流れる。

生成 AI の出力自体もリスクを持つ。誤送信しやすい文面、もっともらしいが誤った要約、権利侵害の可能性がある文章、攻撃的な返信案などは、利用者が AI が言ったから と採用すると事故になる。だから第26章では、AI のリスクを 利用者教育 だけでなく、使わせる範囲の設計 と 出力確認が必要な用途の明示 で受ける方がよいと書きたい。

## 小さく始めて継続監視する

生成 AI は、いきなり全社展開しない方がよい。理由は、効果の測定より先に、共有や権限やログの崩れが出やすいからである。Microsoft の oversharing blueprint も、試行導入、展開、運用の段階で考える。第26章でも、この進め方を採った方がよい。

最初の試行導入は、低リスクで効果が見えやすい用途が向く。たとえば次のようなものだ。

- 公開済み資料の要約
- 社内規程や FAQ の検索補助
- 会議メモの下書き
- 定型メールの下書き
- 情シス内部のナレッジ整理

逆に、最初から避けたいのは次のような用途である。

- 顧客個人情報を含む問い合わせ対応
- 人事評価や懲戒の文案
- 契約確定文の自動生成
- 外部システムを更新する agent
- 権限付与や送金に関わる自動判断

試行導入では、対象部門、対象データ、対象サービス、管理者、評価基準、停止条件を決めたい。評価基準も、使った人数 だけでは弱い。次のようなものが実務的である。

- どの用途で時間短縮が出たか
- ルール違反入力が起きていないか
- ログで追えるか
- 共有事故や過剰共有が出ていないか
- 利用者が出力を人手確認しているか
- 続ける価値があるか

ここで重要なのは、利用率だけで成功判定しないことだ。使われていても、機密が混ざる、監査できない、権限が広すぎるなら、その導入は未完成である。

## 通常時の例と、崩れた時の例

通常時の例では、情シスと経営企画の少人数で、社内規程、情報システム手順、公開済み FAQ だけを対象にした AI 検索を試行導入する。対象データは個人情報や契約情報を含まないものに絞る。共有範囲は試行導入メンバーだけにし、外部

GPT、actions、apps は無効にする。ログ取得可否を確認し、1 か月ごとに入力違反、誤回答、利用満足度、時間短縮をレビューする。これなら、第26章で言いたい 限定部門 限定データ 限定機能 がそろう。

別の通常時の例では、Microsoft 365 Copilot を営業支援チームの 5 人へ試行導入する。ただし、その前に OneDrive と SharePoint の共有設定を点検し、誰でも閲覧 や広すぎる共有リンクを是正する。Copilot は 今まで見えなかった情報を新たに解放する のではなく 既に見える情報を見つけやすくする ので、先にデータ側を整える。導入後は prompts や outputs の監査導線も確認する。これで、AI 導入と権限整理が一つの流れになる。

崩れた時の例では、社員が個人向け生成 AI で議事録要約を始め、次第に顧客名や商談内容も入力し始める。会社側には承認済みサービス一覧も入力禁止情報もなく、何がどこへ入ったか追えない。別の例では、社内用 GPT を一人の担当者が作り、actions で外部 SaaS と接続していたが、退職時に所有者整理がされず、どのデータがどこへ送られていたか分からなくなる。さらに別の例では、Gem sharing を許したまま Drive の外部共有も広く、社内用のつもりの Gem が外へ共有される。問題は AI を使ったことではなく、使う条件 を決めずに広げたことである。

## 最低限ここまではやる

第26章の内容を一度に全部整えなくてもよい。だが、次の五つは早めに持ちたい。

1. 承認済みサービスと未承認サービスを分ける
2. 入力禁止情報と条件付き許可情報を一枚で定める
3. 外部 GPT、actions、apps、connectors の扱いを決める
4. ログ、保持期間、監査手段を確認する

5. 低リスク用途だけで試行導入を始める

この五つがあるだけで、生成 AI はかなり管理しやすくなる。生成 AI とは、便利なチャットを増やすことではない。会社がどの条件で使ってよいかを決め、その条件の中で価値を出すことである。

## 今日、今週、後でやること

今日やることは、いま社内で使われている生成 AI サービスを洗い出し、承認済み 未承認 不明 に分けることである。あわせて、個人情報、顧客情報、秘密情報、認証情報を入力禁止情報として仮置きしたい。

今週やることは、承認済みサービスを一つ選び、対象ユーザー、対象用途、入力可能情報、外部接続可否、ログ取得可否、保持期間を一枚で整理することである。その上で、低リスク用途に絞った試行導入を設計する。

後でやることは、カスタム GPT や Gem や社内検索型 AI を業務資産として台帳化し、所有者、編集者、共有範囲、接続先、停止手順、退職時引き継ぎまで整えることである。ここまでできると、第26章の AI ガバナンスは 禁止一覧 ではなく、使える形で守る運用になる。

第27章

# ひとり情シスの働き方、引き継ぎ、次の一手

本書の最後に置きたいのは、ひとり情シスは大変だが頑張ろう という話ではない。むしろ逆である。ひとり情シスの仕事は、頑張るほど一人に集まりやすい。問い合わせも、管理者権限も、契約更新も、ベンダー連絡も、障害初動も、社内の相談も、最後には あの人しか分からない に寄っていく。これは能力の高さの証明にも見えるが、会社にとっては停止リスクでもある。

第27章で強調したいのは、ひとり情シスの完成形は 全部知っている人 になることではないという点である。本当に目指したいのは、自分が不在でも最低限回る状態を作ることだ。長期休暇でも、急病でも、異動でも、退職でも、会社に知識と運営が残る。その状態を作るには、仕事の持ち方、時間の使い方、引き継ぎ資料、代替体制、次に進める改善テーマを意識的に設計し直す必要がある。

## 何を自分で持ち、何を任せるか

ひとり情シスが最初に見直したいのは、自分が全部やる前提 で仕事を抱えていないかである。ここで大事なのは、仕事を 作業 の単位だけで見ないことだ。NISTの NICE Framework が示す通り、仕事は role や task のまとまりで見る方が整理しやすい。つまり、PC を配る 権限を付ける SaaS を契約する を別々の雑用として持つのではなく、どこに判断責任があり、どこが単純実行で、どこが他者へ渡せるのかを先に分ける。

実務では、次の四つに分けるとよい。

- 自分で決める仕事

- 他者に実行してもらう仕事
- 自動化する仕事
- やめる仕事

自分で決める仕事 には、たとえば次が入る。

- 重要システムの導入判断
- 高権限の付与基準
- 外部委託の責任分界
- 障害や重大インシデント時のエスカレーション判断
- 経営へ上げるべきリスクや予算の説明

ここは、ひとり情シスが持つべきである。単なる作業ではなく、会社としての説明責任があるからだ。

一方で、他者に実行してもらう仕事 は意外に多い。たとえば、入社者情報の回収は人事、アプリ利用申請の妥当性判断は部門長、経費や請求書の一次確認は経理、保守作業はベンダー、専門的なネットワーク調査やフォレンジックは外部専門家へ渡せる。ここでのポイントは、全部外に出す ことではない。判断は社内に残し、実行を渡すことである。

さらに、自動化する仕事 と やめる仕事 も同じくらい重要である。毎回同じ内容のアカウント発行依頼を手入力しているなら、申請フォームと定型化でかなり減る。毎月見ていない報告書を手で作っているなら、その仕事はやめた方がよい。ひとり情シスでは、やるべきか を見直さずに どう早くやるか だけ考えると、忙しさが固定化する。

ここで一つ意識したいのは、自分しかできない仕事 と 自分しかやっていない仕事 は違うという点である。後者は、仕組みを作っていないだけのことが多い。第27章では、この差を見抜けるようになることが重要である。

## 忙しさに飲まれない仕事設計

ひとり情シスの忙しさは、件数の多さだけでは決まらない。いつでも割り込まれる　どこから依頼が来るか決まっていない　同じ種類の仕事を毎回ばらばらに処理する　期限管理が頭の中にしかない　という設計だと、件数がそれほど多くなくても疲弊する。

したがって、第27章では働き方を気分の問題ではなく、仕事設計の問題として見たい。最初に整えたいのは、依頼の入口を一つに寄せることである。第2章や第16章でも触れた通り、依頼窓口がチャット直投げ、口頭、個人メール、会議ついでに散っていると、優先順位も証跡も崩れる。ひとり情シスほど、入口は一つに寄せた方がよい。

次に、仕事を一定のリズムで持つ。たとえば、次のように切る。

- 毎日: 新規依頼の一次確認、緊急度判定
- 毎週: 滞留案件と未完了の変更作業の見直し
- 毎月: 契約更新、ライセンス棚卸し、主要ログ確認
- 四半期: 権限棚卸し、ベンダーレビュー、改善テーマの進捗確認

これを決めておくと、思い出した時にやる　から抜けられる。ひとり情シスでは、作業力より、忘れにくい構造を作る方が効く。

また、似た作業はまとめて処理した方がよい。新規入社の端末準備を一件ずつ都度対応するより、毎週決まったタイミングでまとめる方が安定する。ソフトウェア申請、共有フォルダ権限、貸与品の更新も同じである。まとめ処理にするだけで、割り込みはかなり減る。

さらに、集中して考える時間　を守る必要がある。導入判断、変更計画、障害レビュー、規程見直しのような仕事は、空き時間の切れ端では進まない。毎日一時間でもよいので、連絡を返す時間と、考える時間を分ける方がよい。忙しい会社ほど、ここを守らないと、いつまでも運用の火消しだけで終わる。

忙しさは美徳ではない。忙しい状態が続いているなら、努力不足より先に、入口、標準化、頻度、例外の扱いに設計不良がないかを見る方がよい。

## 引き継ぎ資料と後継者不在リスク

ひとり情シスで最も危ないのは、後継者がいないことそのものではない。いま自分が止まったら、会社がどこを見ればいいか分からない 状態である。したがって、後継者採用の前に、最低限の引き継ぎ束を作る必要がある。

ここでよくある失敗は、完全版マニュアルを目指して何もできないことだ。だが、最初に必要なのは全システムの詳説ではない。緊急時に見れば最低限回せる一式である。

最低限の引き継ぎ資料には、次を入れたい。

- 主要サービス一覧と重要度
- 管理画面や契約の正式名称
- 管理者権限の所在
- 緊急用アカウントの有無と利用条件
- 秘密情報や復旧手段の保管場所
- ベンダー、保守、リセラー、ISP の連絡先
- ドメイン、証明書、回線、SaaS の更新期限
- バックアップと復元テストの状況
- 日次、週次、月次で見るべきもの

- 障害やインシデント時の最初の連絡先

この一式があるだけで、長期休暇前の安心感はかなり変わる。

管理者権限の代替体制は、特に早く整えたい。Google は複数の super admin アカウントを別々の個人へ割り当てることを勧めている。Microsoft も、クラウド専用の緊急用アカウントを二つ持つことを勧めている。ここから分かるのは、管理者権限が一人にしかない 状態は、小さな会社でも避けるべきだということである。

また、管理者アカウントは日常利用アカウントと分けた方がよい。普段のメールや Web 閲覧に高権限アカウントを使うと、誤操作もフィッシングも一気に重くなる。第8章でも扱ったが、第27章ではこれを セキュリティの話 にとどめず、引き継ぎしやすさ の話として見る。高権限が用途ごとに分かれていれば、後から見た時に構造が分かりやすい。

緊急用アカウントや回復情報の保管も、一人の記憶や個人端末に置かない方がよい。たとえば、管理部門の責任者がアクセスできる安全な場所に、利用条件と連絡フローつきで保管する。利用したら必ず記録する。高優先度アラートや監査ログが取れるなら、そこまで整える。重要なのは、誰でも使える ではなく 必要な時に認可された人が使える 状態にすることである。

そして、後継者不在リスクを考える時は、次の情シス担当がまだいない ことと、いま誰も代替できない ことを分けて考えたい。前者は採用や体制の問題だが、後者は今日から下げられる運営リスクである。

## 相談相手と外部ネットワークの作り方

ひとり情シスは、一人で実務を回すことはあっても、一人で全部判断するべきではない。にもかかわらず、現場では 困ったら自分が考える が常態化しやすい。これを防ぐには、相談先を事前に設計する必要がある。

まず、社内の相談先を整理する。たとえば、次は分けて持ちたい。

- 予算や契約条件の判断を上げる相手
- 個人情報や労務に関わる相談先
- 業務影響や優先順位を決める部門責任者
- 緊急時に利用停止や対外連絡を判断する経営層

これが決まっていないと、障害やインシデントの時に、技術判断と経営判断が混ざる。

次に、外部の相談先も 誰に何を聞くか まで整理したい。候補はたとえば次である。

- Microsoft や Google のリセラー、販売代理店
- ネットワーク保守や回線事業者
- セキュリティ事故時に支援を頼める専門会社
- 個人情報や契約確認を相談できる法務や社労士
- 請求や支払い影響を見られる経理担当

ここで大事なのは、連絡先だけでなく、契約番号、受付時間、一次切り分けに必要な情報、緊急時の連絡順まで残すことである。電話番号だけあっても、深夜障害で契約名義が分からなければ進まない。

さらに、同じ立場の人と話せる外部ネットワークも持っておく方がよい。必ずしも formal な組織である必要はない。リセラーの担当者、地域の IT コミュニティ、同業の管理部門、情報交換できる社外の知人でもよい。重要なのは、自分の会社の常識だけで判断しない ための比較軸を持つことである。

相談は、困ってから探すと遅い。第27章では、相談先一覧も引き継ぎ資料の一部として持つべきだと書きたい。

## 次に整えるべきテーマを選ぶ

本書をここまで読むと、直したい場所はたくさん見つかる。だが、ひとり情シスが全部を同時に進めると、だいたい崩れる。大事なのは、何が抜けているか を知ることより、次にどれから埋めるか を決めることである。

ここで役立つのが、現状と目標の差分で考えるやり方である。NIST CSF の Current Profile と Target Profile の考え方を使えば、いまはどうか 次にどこまで持っていきたいか を見比べられる。CISA の CPG も、中小規模組織では高い効果が見込める少数の重要対策を優先する考え方を示している。つまり、次年度計画は網羅性より順序である。

実務では、次の四つで見ると絞りやすい。

- リスクをどれだけ下げるか
- 不在時でも回る度合いをどれだけ上げるか
- 毎月の工数をどれだけ減らせるか
- 属人化をどれだけ減らせるか

たとえば、候補が十個あったとしても、次の 6 か月から 1 年では三つくらいに絞った方がよい。たとえば次のような組み合わせである。

- 管理者権限と緊急用アカウントの代替体制を整える
- 端末、アカウント、契約の台帳を一本化する
- 問い合わせの入口と定型依頼を標準化する

これなら、リスク、継続性、工数削減の三方向を同時に押せる。

逆に、よくない進め方は、思いついた改善を全部未整理のまま積む ことである。項目が 30 個あっても、それは計画ではない。やる順、終わりの条件、担当、依存関係がない限り、ただの不安一覧である。第27章では、テーマは少なく、条件は具体的に を強めに置きたい。

## 通常時の例と、崩れた時の例

通常時の例では、長期休暇の二週間前に、ひとり情シスが 緊急時だけ見る引き継ぎ束 を一つにまとめる。そこには、主要サービス一覧、管理者権限の場所、緊急用アカウント、ベンダー連絡先、更新期限、障害時の最初の連絡先が入っている。Google Workspace では super admin が複数あり、Microsoft では緊急用アカウントが整備され、利用時の記録ルールもある。経営層と管理部門には、どの条件でどこへ連絡するかも伝わっている。これなら、担当者が不在でも会社は止まりにくい。

別の通常時の例では、今年やりたいことを十個書き出したあと、高権限の代替体制 契約と更新期限の見える化 問い合わせの標準化 の三つに絞る。四半期ごとに進捗を見る。進めないテーマは捨てる。これにより、何もかも少しずつ未完了 を避けられる。

崩れた時の例では、管理者権限が一人しかなく、そのアカウントを日常のメールにも使っている。バックアップコードも回復情報も本人しか知らない。別の例では、ドメイン管理が昔の制作会社の個人アドレスに紐づいたままで、証明書更新もその人しか分からない。さらに別の例では、契約更新日と解約期限が個人カレンダーにしかなく、退職時に何が自動更新されるか誰も把握できない。どれも問題は専門知識不足ではない。個人依存を運営へ変えていないことにある。

## 最低限ここまではやる

第27章の内容を一度に全部整えなくてもよい。だが、次の五つは早めに持ちたい。

1. 仕事を 自分で決める 他者に実行してもらう 自動化する やめる に分ける
2. 管理者権限の代替経路と緊急用アカウントを整える
3. 緊急時に見る引き継ぎ束を一つ作る
4. 相談先、ベンダー、更新期限を一覧化する
5. 次の 6 か月から 1 年の改善テーマを三つ以内に絞る

この五つがあるだけで、ひとり情シスはかなり続けやすくなる。大事なのは、全部を自分で抱えたまま強くなることではない。会社に知識と運営を残し、自分しか回せない状態を減らすことである。

## 今日、今週、後でやること

今日やることは、いま自分が抱えている仕事を書き出し、判断 実行 自動化候補 やめる候補 に分けることである。あわせて、管理者権限が一人しかないサービスがないかを確認したい。

今週やることは、引き継ぎ束の初版を作ることである。完璧でなくてよい。主要サービス、管理者権限、緊急連絡先、更新期限、秘密情報の保管場所だけでも先に一か所へ集めたい。

後でやることは、次の 6 か月から 1 年の改善テーマを三つに絞り、四半期ごとの見直し日を決めることである。ここまでできると、第27章の終わり方は 頑張り続ける ではなく、続けられる形に変える になる。

付録

# すぐ使えるひな型とチェックリスト

台帳、入退社運用、有事初動をそのまま実務へ落とし込む。

全3本

付録A

# 台帳ひな型

付録Aでは、本書で繰り返し登場した台帳の最小ひな型をまとめる。最初から完璧な CMDB を作る必要はない。重要なのは、何があるか 誰が持つか いつ見直すか が後から追えることである。

## A-1 サービス台帳

| 項目 | 何を書くか |
|---|---|
| サービス名 | 正式名称、略称 |
| 用途 | 何の業務で使うか |
| 重要度 | 高、中、低 |
| 管理者 | 主担当、代替担当 |
| 提供元 | ベンダー名、契約窓口 |
| 契約形態 | 月額、年額、自動更新など |
| 更新期限 | 契約更新日、解約期限 |
| 認証方式 | ID/PW、SSO、MFA |
| 保管データ | 個人情報、契約情報、ログなど |
| バックアップ | 取得有無、復元方法 |
| 障害時連絡先 | ベンダー、社内判断者 |

## A-2 アカウント台帳

| 項目 | 何を書くか |
|---|---|
| 対象サービス | Microsoft 365、Google Workspace など |
| アカウント種別 | 個人、共有、管理者、サービス |
| 所有者 | 利用者名、部門 |
| 権限 | 一般、管理者、限定管理者など |
| MFA | 有効、無効、例外理由 |
| 代替管理者 | 緊急時に使える人 |
| 棚卸し日 | 最終見直し日 |
| 状態 | 有効、停止、退職処理中、削除済み |

## A-3 端末台帳

| 項目 | 何を書くか |
|---|---|
| 資産番号 | 社内管理番号 |
| 種別 | ノート PC、スマホ、タブレット |
| 利用者 | 氏名、部門 |
| OS | Windows、macOS、iOS、Android |
| 管理方法 | MDM、手動、未管理 |
| 配布日 | 貸与日 |
| 保証期限 | 保守、保証、リース期限 |
| 状態 | 利用中、予備、修理中、退役 |
| 備考 | 紛失、交換履歴など |

## A-4 契約台帳

| 項目 | 何を書くか |
|---|---|
| 契約名 | サービス名、保守契約名 |
| 契約先 | ベンダー、販売代理店 |
| 契約番号 | 見積番号、注文番号、契約番号 |
| 金額 | 月額、年額、従量 |
| 更新条件 | 自動更新、都度更新 |
| 解約期限 | いつまでに止める必要があるか |
| 支払方法 | 請求書、カード、口座振替 |
| 社内担当 | 契約管理者、経理窓口 |
| 関連サービス | どのシステムに紐づくか |

## A-5 連絡先台帳

| 項目 | 何を書くか |
|---|---|
| 相手先 | ベンダー、ISP、保守会社、法務など |
| 目的 | 障害、契約、請求、法務相談など |
| 連絡手段 | メール、電話、ポータル |
| 受付時間 | 平日、24時間など |
| 契約情報 | 契約番号、顧客番号 |
| 連絡条件 | どの状況で連絡するか |
| 社内判断者 | 誰の承認で連絡するか |

## A-6 最小構成で先に持つべき台帳

最初から全部を細かく作らなくてよい。まずは次の四つを優先するとよい。

- サービス台帳
- アカウント台帳
- 端末台帳
- 契約台帳

この四つがそろうだけで、入退社、障害、更新、引き継ぎの多くがかなり見えやすくなる。

付録B

# 入社、異動、退職チェックリスト

付録Bでは、第9章を実務で使いやすい形へ落とすために、入社、異動、退職の最小チェックリストをまとめる。会社ごとに追加項目はあるが、まずは抜けやすい場所を塞ぐことを優先する。

## B-1 入社前

- □ 氏名、所属、役職、開始日を確定した
- □ 必要なアカウントを洗い出した
- □ 標準端末を確保した
- □ ライセンスの空きと追加要否を確認した
- □ 所属部門へ必要権限の承認者を確認した
- □ メールアドレス、表示名、命名規則を確認した
- □ 端末配布日と初回ログイン方法を決めた

## B-2 入社初日

- □ 本人確認をした
- □

初期パスワード変更または初回サインインを完了した

- ☐ MFA 登録を完了した
- ☐ 必要最小限のグループと権限だけ付与した
- ☐ 端末の受領確認を取った
- ☐ 問い合わせ窓口と基本ルールを案内した
- ☐ 共有アカウントや代表アドレスの扱いを説明した

## B-3 入社後1週間以内

- ☐ 利用していないアカウントやライセンスがないか確認した
- ☐ 過剰権限が付いていないか確認した
- ☐ 端末やスマホの管理状態を確認した
- ☐ 必要な共有先だけ見えているかを確認した

## B-4 異動時

- ☐ 異動日と新旧所属を確認した
- ☐ 新しい業務に必要な権限を承認ベースで付与した
- ☐ 旧所属で不要になった権限を剥奪した

- ☐ 共有フォルダ、メーリングリスト、チャネル参加を見直した
- ☐ 管理者権限や例外設定が残っていないか確認した
- ☐ 利用端末や貸与品の変更有無を確認した

## B-5 退職前

- ☐ 最終出社日と雇用終了日を確認した
- ☐ 当日停止が必要か、引き継ぎ後停止でよいかを決めた
- ☐ データ引き継ぎ先を決めた
- ☐ 共有アカウント、代表アドレス、外部サービスの所有を確認した
- ☐ 貸与品の回収方法を決めた
- ☐ 委託先や外部ゲストの終了時は接続先も洗い出した

## B-6 退職当日

- ☐ サインインを遮断した
- ☐ セッションやトークンを失効した
- ☐ MFA 手段や回復手段を無効化した
- ☐ 管理者権限を剥奪した

- □
  VPN、Wi-Fi 証明書、MDM 登録を停止した
- □
  貸与 PC、スマホ、IC カード、物理鍵を回収した

## B-7 退職後

- □
  メール転送や自動返信の要否を決めた
- □
  ファイル所有権や共有設定を整理した
- □
  アーカイブ、停止、削除のどれにするか決めた
- □
  ライセンスを回収した
- □
  契約、請求、外部サービスに本人名義のものが残っていないか確認した
- □
  台帳と棚卸し記録を更新した

## B-8 迷った時の優先順位

迷った時は、次の順で考えるとよい。

1. まずサインインと権限を止める
2. 次にデータと所有権を引き継ぐ
3. 最後にライセンス、アーカイブ、削除を整理する

退職処理で危ないのは、削除を急ぎすぎて証跡やデータを失うことである。遮断と失効を先に行い、その後に整理する方が事故が少ない。

付録C

# 障害、インシデント初動チェックリスト

付録Cでは、第21章と第22章の初動を、現場で見返しやすい最小チェックリストへ落とす。ここで重要なのは、すぐ直そうとする前に、影響、連絡、証跡、切り分けを押さえることである。

## C-1 まず共通で確認すること

- ☐ 何が起きているかを一文で書いた
- ☐ 発生時刻または発覚時刻を記録した
- ☐ 影響範囲を確認した
- ☐ 影響を受ける業務と利用者を確認した
- ☐ 単一障害か広域障害かを見た
- ☐ 一つのチケットまたは記録場所を決めた
- ☐ 連絡担当を決めた

## C-2 通常の障害初動

- ☐ 回線、認証、DNS、クラウド障害情報を確認した
- ☐

直前の変更や更新を確認した

- □ 影響サービス、端末、拠点を切り分けた
- □ 利用者向けの一次連絡を出した
- □ 切り戻しが必要かを判断した
- □ 暫定復旧でよいか、本復旧が必要かを決めた

## C-3 セキュリティ事故の疑いがある時

- □ 不正アクセス、情報漏えい、ランサムウェアの可能性を切り分けた
- □ すぐ止めるべきアカウントや端末を隔離した
- □ ログ、メール、画面、アラートを保全した
- □ 安易に削除や再起動をして証跡を消していないか確認した
- □ 経営、法務、個人情報保護の判断者へ連絡した
- □ 外部専門家やベンダーへ支援要否を判断した

## C-4 利用者向け一次連絡

- □ 何が起きているかを簡潔に書いた
- □ 影響範囲を書いた

- □ いま分かっていることと未確定事項を分けて書いた
- □ 利用者に止めてほしい行動を書いた
- □ 次の更新予定時刻を書いた

## C-5 外部連絡前に確認すること

- □ 契約番号、顧客番号、テナント情報を手元に置いた
- □ 影響範囲と発生時刻を整理した
- □ 何を依頼したいかを一文で言える状態にした
- □ 社内承認が必要な連絡か確認した

## C-6 復旧後に必ずやること

- □ 復旧時刻を記録した
- □ 根本原因と直接原因を分けて整理した
- □ 暫定対応と恒久対応を分けた
- □ 再発防止の担当者と期限を決めた
- □ 責任追及を目的にしない振り返りを行った
- □ 手順書、台帳、監視、設定を更新した

## C-7 初動で避けたい行動

- □ 記録せずに個人チャットだけで進める
- □ 影響範囲が分からないまま 復旧しました と言う
- □ 証跡保全前に削除や初期化をする
- □ 連絡窓口が複数に分かれて情報がずれる
- □ 直った後にレビューをやらず終える

障害でもインシデントでも、初動の質は どれだけ早く全部分かるか ではなく、分かっていないことを含めて整理しながら進められるか で決まる。

# 奥付

- 書名: ひとり情シス大全
- 状態: 本文ドラフト
- 作成日: 2026年3月21日
- 構成: 全27章、付録3本
- 備考: 製品名、管理画面名、制度、公式ガイドの版は今後の更新にあわせて見直す前提

この PDF は、本文構成と内容確認のための制作途中版である。