MIXI

2022卒セキュリティ研修

セキュリティ室

2022年5月10日

講師の自己紹介

小澤　拓海です。

- 開発本部セキュリティ室所属
- 前職
  - Webアプリの脆弱性診断関係の仕事を約4年
- 今
  - IaaS監視/啓発/教育/診断に関係することなど色々
- プライベート
  - 絵
  - カラオケ

研修の目的

MIXI
CONFIDENTIAL

目的	● 情報セキュリティの定義を知る。 ● なぜ情報セキュリティが必要なのかを知る。 ● Webアプリケーションの著名な脆弱性の原理と対策方法を知る。 ● スマホアプリのセキュリティ上の注意点を知る。 ● エンジニアとして必要とされる情報セキュリティの観点を知る。 ● **セキュリティに興味をもってもらう。**

アジェンダ

MIXI
CONFIDENTIAL

Day 1

情報セキュリティの基本を知ろう

ミクシィ社で取り組んでいる情報セキュリティ対策を知ろう

セキュアに開発するにはどうしたらいいか知ろう with ハンズオン

Day 2

脆弱性対策演習

アジェンダ

Day 1

情報セキュリティの基本を知ろう

ミクシィ社で取り組んでいる情報セキュリティ対策を知ろう

セキュアに開発するにはどうしたらいいか知ろう with ハンズオン

Day 2

脆弱性対策演習

1日目は基礎知識習得と、実際に試すことで攻撃手法を知ることを主な目的とします。

アジェンダ

MIXI
CONFIDENTIAL

Day 1

情報セキュリティの基本を知ろう

ミクシィ社で取り組んでいる情報セキュリティ対策を知ろう

セキュアに開発するにはどうしたらいいか知ろう with ハンズオン

Day 2

脆弱性対策演習

2日目は「対策」にフォーカスして演習を進めていきます。

本研修では
実際に悪用すると不正アクセス禁止法に
抵触する内容を含んでいます。

悪用厳禁！

タイムテーブル

MIXI
CONFIDENTIAL

※適宜休憩/昼休みを挟みつつ。
※順調に行けば早めに終わる(はず)。

時間	内容	種別
10:30–11:00	情報セキュリティの基本を知ろう	座学
11:00–11:30	ミクシィ社で取り組んでいる情報セキュリティ対策を知ろう	座学
11:30–16:00	（サーバ側） メジャーな脆弱性を知ろう	座学 + ハンズオン
16:00–16:20	（サーバ側） 外部ライブラリ等の脆弱性情報の見方を知ろう	座学
16:20–17:40	セキュアに開発するにはどうしたらいいか知ろう （サーバ側） アクセス制御の考え方とやり方を知ろう	座学 + ハンズオン
17:40–18:30	（クライアント側） クライアント側のセキュリティリスクを知ろう	座学

第一章
情報セキュリティの基本を知ろう

情報セキュリティとは

MIXI
CONFIDENTIAL

下記の３要素のことを言う。

- 情報セキュリティにおいて「セキュアである」とは、下記３つのうち**ひとつも**侵害されていない状態のこと。
  - 機密性
  - 完全性
  - 可用性

情報セキュリティとは

機密性とは

- 許可された者だけが情報にアクセスできること。

機密性が侵害されている例

- あるECサイトにて、アクセス権の検証に不備があり、他ユーザの購入履歴を参照できる状態になっていた。

情報セキュリティとは

MIXI
CONFIDENTIAL

完全性とは

- 情報が正確であること。改ざんされたり破壊されたりしていないこと。

完全性が侵害されている例

- 会社のホームページが不正アクセスされ、見た目やリンクが改ざんされている。

情報セキュリティとは

MIXI
CONFIDENTIAL

可用性とは

- 情報が必要なとき、いつでもそれにアクセスできること。

可用性が侵害されている例

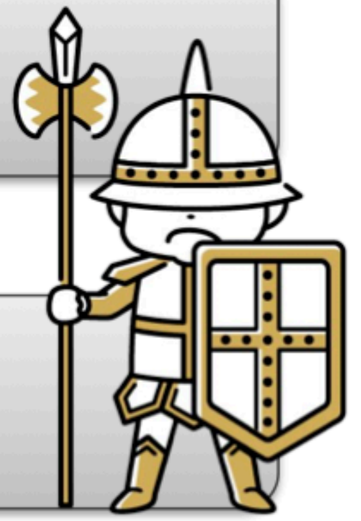
- 運営しているWebサービスに対し、負荷の大きなリクエストを大量に実行され、耐え切れずサーバがダウンしてしまっている。

何のために情報セキュリティを学ぶのか

サービスを利用する「利用者」と、それを提供する「自分たち」を守るためです。

利用者を守る

自分たちを守る



情報セキュリティを守れず、事故が起きると……

MIXI
CONFIDENTIAL

利用者は……

実害的なダメージ

自分や自分の家族等の個人情報が漏洩する。
課金していたゲームの資産が無駄になる。
必要だったサービスが受けられない。
etc

自分たちは……

実害的なダメージ

賠償や訴訟に発展することがある。
業務自体が停止する。売上への悪影響。

信頼的なダメージ

「ミクシィ」というブランドへの信頼が低下する。
ユーザが今後ミクシィのサービスを利用したくなくなる。
コラボ等のビジネスチャンスを信頼失墜で失う。

言葉の整理　脅威/脆弱性/リスクとは

- 脅威
  - 事故の原因となるもののこと。クラッカー(攻撃者)など。
- 脆弱性
  - 脅威によって実際に事故が起こりやすい性質のこと。
- リスク
  - 脅威や脆弱性等を考慮したうえで、
    どれくらい事故が起きそうか、またどれくらいの被害になりそうかの度合い。

Confidential

Confidential

Confidential

Confidential

Confidential

Confidential

Confidential

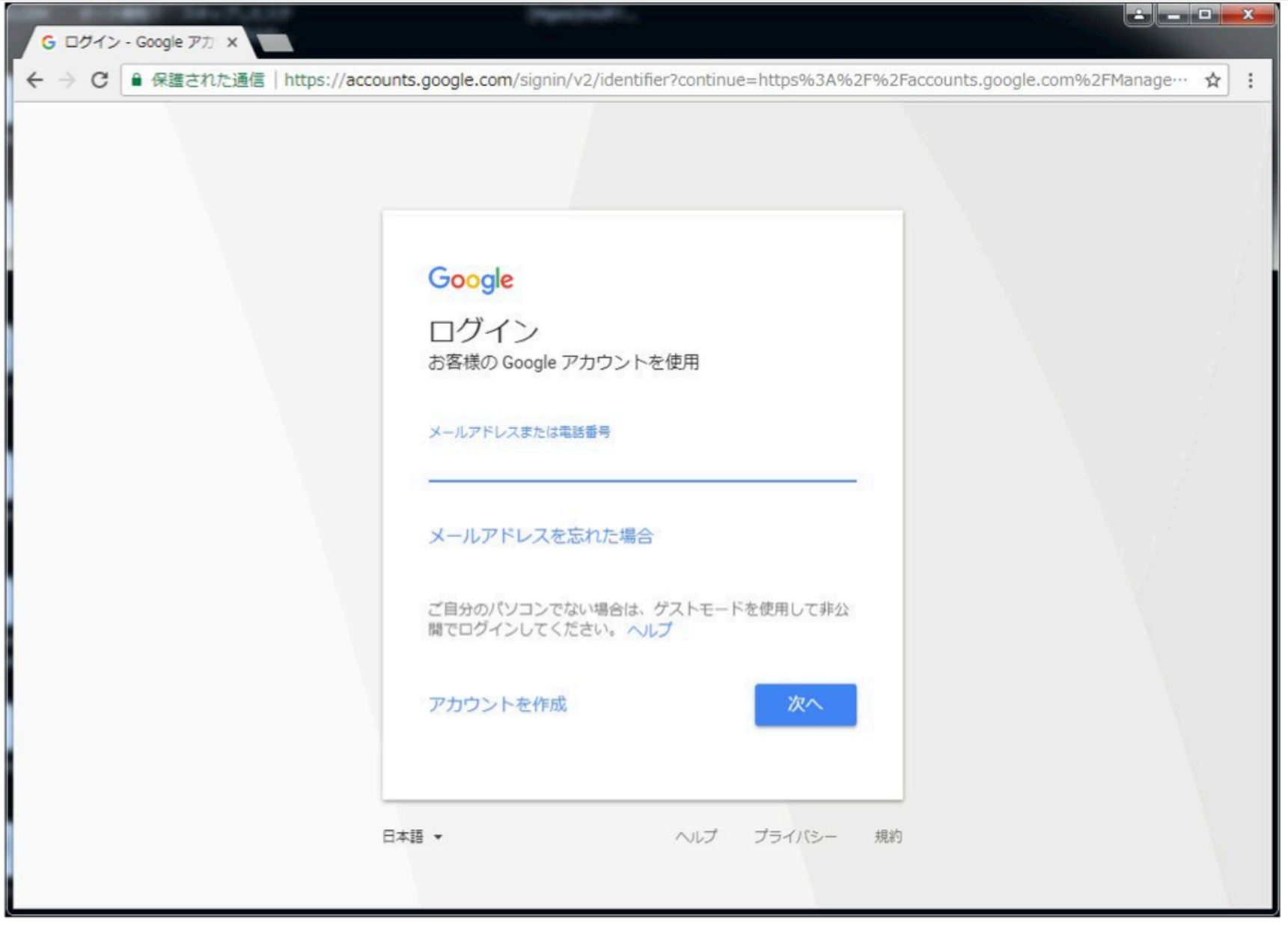
MIXI

間違い探し！

偽物


ログイン - Google アカ
保護された通信 | https://accounts.google-acc.com/signin/v2/identifier?passive=1209600&continue=https%3A%2F%2Faccounts.…
Google
ログイン
お客様の Google アカウントを使用
メールアドレスまたは電話番号
メールアドレスを忘れた場合
ご自分のパソコンでない場合は、ゲストモードを使用して非公開でログインしてください。ヘルプ
アカウントを作成
次へ
日本語
ヘルプ
プライバシー
規約

本物


ログイン - Google アカ
保護された通信 | https://accounts.google.com/signin/v2/identifier?continue=https%3A%2F%2Faccounts.google.com%2FManage…
Google
ログイン
お客様の Google アカウントを使用
メールアドレスまたは電話番号
メールアドレスを忘れた場合
ご自分のパソコンでない場合は、ゲストモードを使用して非公開でログインしてください。ヘルプ
アカウントを作成
次へ
日本語
ヘルプ
プライバシー
規約

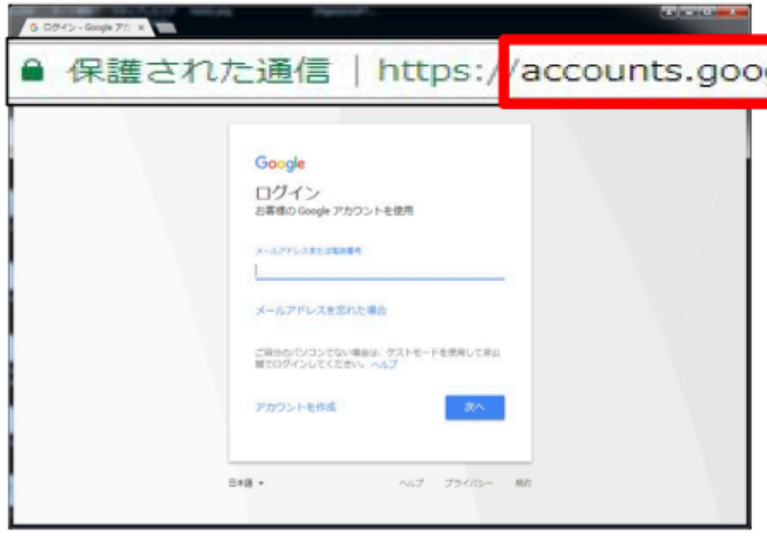
MIXI

お分かりいただけただろうか

違いはコカ

MIXI
CONFIDENTIAL

※偽物

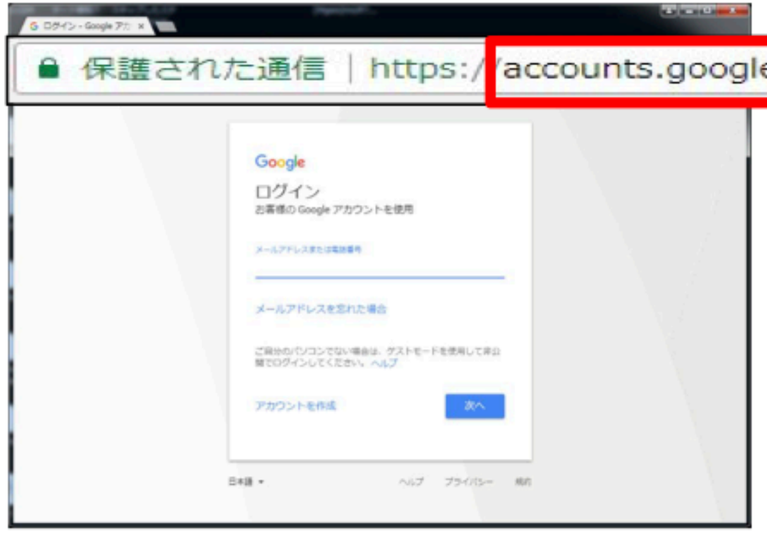


証明書情報

発行先:	google-acc.com
発行者:	Let's Encrypt Authority X3
有効期間	2018/ 03/ **20** から 2018/ 06/ 18

無料で有名な
Let's Encrypt

※本物



証明書情報

発行先:	*.google.com
発行者:	Google Internet Authority G2
有効期間	2018/ 03/ **08** から 2018/ 05/ 31

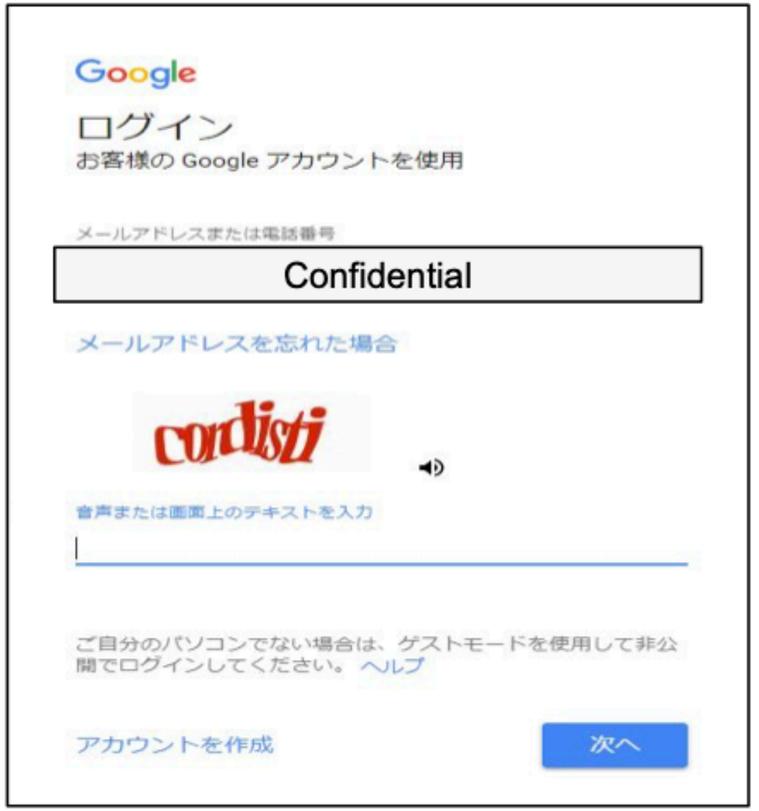
フィッシングサイトでログインを試みるとどうなるか

- 裏で本物のGoogleログインを試行しているため、
  認証の成功/失敗も本物と同様に再現されている。

実在しないID

実在するID

ご丁寧にbot対策まで



安全に配慮し調査＆キャプチャを行っています！

怪しいサイトに興味本位でアクセスしないように

Confidential

事例教訓のまとめ（全般的教訓）

MIXI
CONFIDENTIAL

- 情報セキュリティ事故が発生するとサービス提供者もユーザも不幸になる。

- アプリだけでなく多角的にセキュリティの底上げが必要。
  - とはいえ何もかも完璧にとはいかないのでリスクの高い部分から。

- 事故が起きてしまっても迅速かつ落ち着いた対処する。
  - 事故が起きたら　→ Confidential
  - 不安だったら　→ Confidential

バグと一緒で脆弱性や事故は対策しても起きる時は起きるので、
先人たちの知恵を借りながら出来ることをやっていきましょう。

事例教訓のまとめ（エンジニア的教訓）

MIXI
CONFIDENTIAL

- 認証(本人確認)はシステムの守りの根幹。
  - 何をもって本人と認めるかの設計は慎重に判断。
- フィッシングと見分け辛いので企業サービスでは Let's Encrypt は避けたい。
  - Let's Encrypt自体が悪いわけではなく、否定するわけではない。
  - 無料でお手軽なのでフィッシングで利用しやすいのも事実ではある。
- 社内ネットワークだからといって過度に安心しない。
  - stgやCIツールも注意。
- 認証や権限チェックや棚卸しをしっかり行う。
  - ファイルもシステムも。
- 自社サービスのインフラのパッチ適用を行う。
  - OS、ミドル、パッケージ、etc。
- ロギングはできるだけ行う。
  - アクセスログや操作ログ。
- 脆弱性対策を行う。
  - セキュアに作る。
  - 作った後は脆弱性診断で見つける。

参考（初歩だけどありがちな事故の元）

インターネット公開した瞬間やられた！

- 例
  - jenkins
  - テスト環境
- インターネット公開するとすぐスキャン/攻撃が来ると心得る
- 公開するのは必要最低限のサーバ、必要最低限のポートのみにする
- 適切にアクセス許可しましょう
  - 境界防御で守る
  - ゼロトラストで守る

参考（初歩だけどありがちな事故の元）

MIXI
CONFIDENTIAL

Githubにキー載せてた。Privateじゃなかった。(^ω^)

- 例
  - AWSのアクセスキー
  - GCPのサービスアカウントキー
- Publicリポジトリへの公開すると即ご利用開始となるもよう
  - 機密情報（個人情報/非公開キャラのアセットetc）を閲覧されるかも
  - マイニング農場が作られて高額請求されるかも

第一章　情報セキュリティの基本を知ろう　まとめ

MIXI
CONFIDENTIAL

- 情報セキュリティとは
  - 機密性
  - 完全性
  - 可用性
- 情報セキュリティが守られていないと
  - ユーザが困る
  - 自分たちが困る
- 弊社も含め、世の中で事故は後を絶たない
  - 事故は起きるという危機感を持つ
  - 過去の事故例からは教訓を得て活かそう

第二章
ミクシィ社で取り組んでいる
情報セキュリティ対策を知ろう

情報セキュリティに対する脅威には何があるか？

MIXI
CONFIDENTIAL

「情報セキュリティ10大脅威」がひとつの参考になる。
これを見ると、情報セキュリティの脅威には様々な種類/観点があると分かる。

■「情報セキュリティ10大脅威 2022」

NEW：初めてランクインした脅威

昨年順位	個人	順位	組織	昨年順位
2位	フィッシングによる個人情報等の詐取	1位	ランサムウェアによる被害	1位
3位	ネット上の誹謗・中傷・デマ	2位	標的型攻撃による機密情報の窃取	2位
4位	メールやSMS等を使った脅迫・詐欺の手口による金銭要求	3位	サプライチェーンの弱点を悪用した攻撃	4位
5位	クレジットカード情報の不正利用	4位	テレワーク等のニューノーマルな働き方を狙った攻撃	3位
1位	スマホ決済の不正利用	5位	内部不正による情報漏えい	6位
8位	偽警告によるインターネット詐欺	6位	脆弱性対策情報の公開に伴う悪用増加	10位
9位	不正アプリによるスマートフォン利用者への被害	7位	修正プログラムの公開前を狙う攻撃（ゼロデイ攻撃）	NEW
7位	インターネット上のサービスからの個人情報の窃取	8位	ビジネスメール詐欺による金銭被害	5位
6位	インターネットバンキングの不正利用	9位	予期せぬIT基盤の障害に伴う業務停止	7位
10位	インターネット上のサービスへの不正ログイン	10位	不注意による情報漏えい等の被害	9位

引用元：https://www.ipa.go.jp/security/vuln/10threats2022.html, (2022/4/25).

3つのアタックサーフェスについて

MIXI
CONFIDENTIAL

攻撃に狙われる場所は大きく3種類があります。

- 社内システム
  - 会社のPCやネットワーク、全社共通の業務用ツールへの攻撃等
- サービス
  - 各サービスのアプリケーション、AWSやGCP等のインフラ、独自の業務用ツールへの攻撃等
- 人
  - 標的型攻撃等

ミクシィ社ではそれぞれの面での対策を行っています。順番に例をご紹介します。

社内システムの情報セキュリティ対策の例

MIXI
CONFIDENTIAL

- ゼロトラスト
  - システムに対するアクセス制御施策。
  - 詳しく後述。
- EDR
  - Endpoint Detection and Response
  - エンドポイントにおける脅威の監視/検出技術
  - 会社支給PCにて不審な挙動(マルウェアが疑われるファイルの実行等)がみられるとアラートが上がる
    - 変なことしてるとばれますよ！！＾＾

サービスの情報セキュリティ対策の例

MIXI
CONFIDENTIAL

- 脆弱性診断
- IaaS監視

脆弱性診断の紹介

MIXI
CONFIDENTIAL

- 脆弱性診断とは
  - アプリケーションに潜む脆弱性を探す検査。
    - ブラックボックス診断
      - 疑似的な攻撃を仕掛け、挙動を診て脆弱性を探る。
    - ホワイトボックス診断
      - ソースコードを読んで脆弱性を探る。
- セキュリティ室で行っている脆弱性診断関連のサポート
  - 予算やスケジュール感に応じて診断内容（内製/外注）の提案と診断実施
  - 診断の準備や実施にあたり必要になる手続き/環境準備等のサポート

脆弱性診断の紹介

MIXI
CONFIDENTIAL

例えばこんな脆弱性が見つかります。

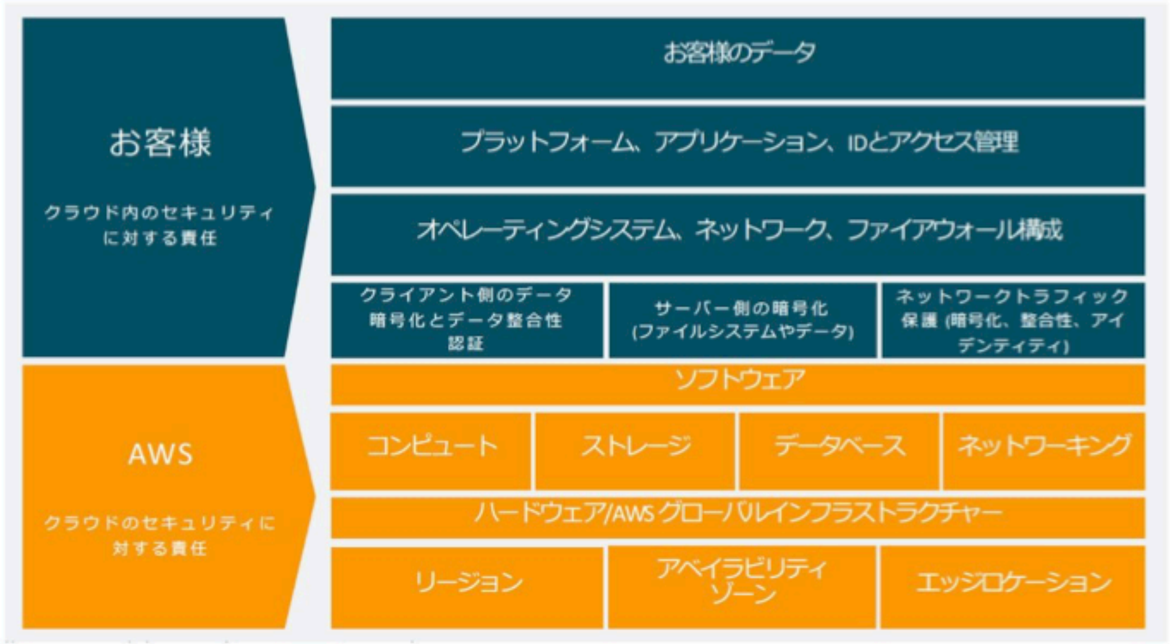
- 他人の登録情報を盗める
- データベースを不正操作できる
- 他人になりすましができる
- 有料ガチャを不正に回し放題
- アイテムを不正に入手できる
- 不正にコードを実行できる

脆弱性診断の公式案内ページは下記です。お気軽にご相談ください。

Confidential

IaaSのセキュリティの前提

- クラウドそのものはIaaS側の責任
- クラウドでしていることはミクシィ側の責任



引用元：https://aws.amazon.com/jp/compliance/shared-responsibility-model/,(2022/4/27)

我々が考慮しなければならないセキュリティ観点

MIXI
CONFIDENTIAL

- ゲストOSより上のあれこれ
  - OS,ミドルウェア,ソフトウェアのバージョン管理
    - バージョン更新やゼロデイのワークアラウンド
  - アプリケーションの脆弱性
- アーキテクチャの設計（設定）
  - 非公開情報が入ったバケットを公開設定してしまっていた
  - 社内用ツールのVMをインターネット公開してしまっていた
- IAMの運用
  - 二要素認証を設定して不正アクセス対策
  - アクセスキーの管理
- 上記に気づける仕組み

ベストプラクティスに従いつつセキュアに運用しましょう

【参考】AWSとGCPのセキュリティベストプラクティス

MIXI
CONFIDENTIAL

AWS

- https://aws.amazon.com/jp/architecture/security-identity-compliance/?cards-all.sort-by=item.additionalFields.sortDate&cards-all.sort-order=desc&awsf.content-type=*all&awsf.methodology=*all

GCP

- https://cloud.google.com/security/best-practices?hl=ja

セキュリティ室で行っているIaaS監視

MIXI
CONFIDENTIAL

- 攻撃が来るとアラートが上がります！
  - 不正なアウトバウンド通信が起こっている
  - マイニングが起こっている

- 設定面での不備があるとアラートが上がります！
  - 二要素認証を設定していないユーザがいる
  - バケットがインターネット公開になっている

セキュリティ室で行っているIaaS監視

MIXI
CONFIDENTIAL

- AWS
  - ご依頼いただければ監視設定をします
    - ただし開発本部の新規アカウント作成フローに則って作成されたアカウントであれば、
      実は作成時点で監視設定を施しています

- GCP
  - 皆さんがミクシィの組織配下にGCPプロジェクトを作成すると実は自動的に監視対象に加わっています
    - ミクシィの配下でない場合は個別にご相談ください！

- その他
  - ミクシィの管理しているドメイン一覧に対して定期的にスクショを撮って、
    目視で内容をチェックしています
    - 右の画像のように、ページ更新を検知すると差分が分かる画像が通知されてきます

詳しくは案内ページで紹介しています！

こんな検出がありました（AWS）

MIXI
CONFIDENTIAL

Confidential

こんな検出がありました（GCP）

MIXI
CONFIDENTIAL

Confidential

こんな検出がありました（スクショ機能）

MIXI
CONFIDENTIAL

Confidential

人の情報セキュリティ対策の例

MIXI
CONFIDENTIAL

- パスワードマネージャ等のソリューション導入
- 啓発/教育活動
  - E-learning
  - 朝会

パスワードマネージャのススメ

MIXI
CONFIDENTIAL

Confidential

Confidential

もしパスワードマネージャを使っていたら防げたか？

MIXI
CONFIDENTIAL

Confidential

MIXI

第三章
セキュアに開発するにはどうしたらいいか知ろう

前提となる考え

- 守らなきゃいけない資産はサーバ側に。
- そのうえでサーバ側を強固に。

サーバ側を強固にするには？

MIXI
CONFIDENTIAL

- 公開するところ
  - 自分たちの作りこむもの（皆さんがコーディングする部分）
    - → 脆弱性を知り、対策する
  - 自分たちでは作りこまないもの（ライブラリやミドルウェア等）
    - → リスクを正しく評価し、必要に応じ対策する
- 公開する必要のないところ
  - → アクセス制御を行って、アクセス自体を絞る

サーバ側を強固にするには？

MIXI
CONFIDENTIAL

- 公開するところ
  - **自分たちの作りこむもの（皆さんがコーディングする部分）**
    - **→　脆弱性を知り、対策する**
  - 自分たちでは作りこまないもの（ライブラリやミドルウェア等）
    - →　リスクを正しく評価し、必要に応じ対策する
- 公開する必要のないところ
  - →　アクセス制御を行って、アクセス自体を絞る

まずはここの話！

作りこんでしまった脆弱性が悪用されたときの被害

MIXI
CONFIDENTIAL

- 個人情報漏洩
- サーバ停止
- 他人を攻撃するための踏み台として利用
- データの破壊、変更、挿入
- Webサイト改ざん
- アカウント乗っ取りによる、なりすまし
- 非公開情報へアクセス
- 不正購入

多岐にわたる

脆弱性を知り、対策するには、、

メジャーどころを網羅するには、OWASP Top10(API/Web)を抑えておくのはベターな選択肢の一つ。

- OWASPとは
  - 「Open Web Application Security Project」の略
  - Webをはじめとするソフトウェアのセキュリティ環境の現状、またセキュアなソフトウェア開発を促進する技術・プロセスに関する情報共有と普及啓発を目的としたプロフェッショナルの集まる、オープンソース・ソフトウェアコミュニティです。
    - 引用元：https://www.owasp.org/index.php/ Japan, (2022/4/27)
- OWASP Top 10とは
  - OWASPがクリティカルだと考える10種類の脆弱性のこと。
  - ことサーバサイド開発においては、API/Webアプリケーションの2種類、計20種類がある。

OWASP Top10の脆弱性たち

MIXI
CONFIDENTIAL

API

- Broken Object Level Authorization
- Broken User Authentication
- Excessive Data Exposure
- Lack of Resources & Rate Limiting
- Broken Function Level Authorization
- Mass Assignment
- Security Misconfiguration
- Injection
- Improper Assets Management
- Insufficient Logging & Monitoring

Web

- Broken Access Control
- Cryptographic Failures
- Injection
- Insecure Design
- Security Misconfiguration
- Vulnerable and Outdated Components
- Identification and authentication failures
- Software and Data Integrity Failures
- Security Logging and Monitoring Failures
- Server-Side Request Forgery (SSRF)

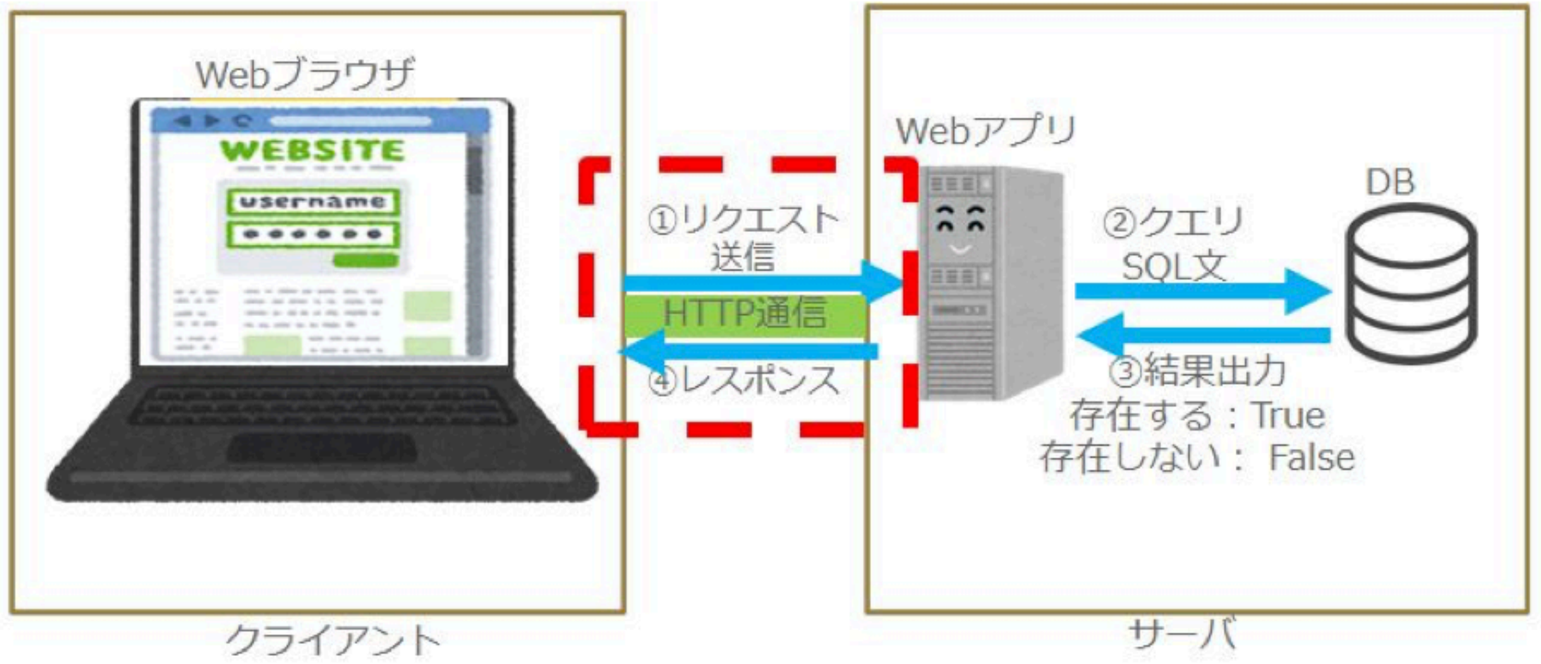
脆弱性を知り、対策する

MIXI
CONFIDENTIAL

Owasp Top 10の脆弱性を、ハッキングを実際に試しながら勉強していきましょう！

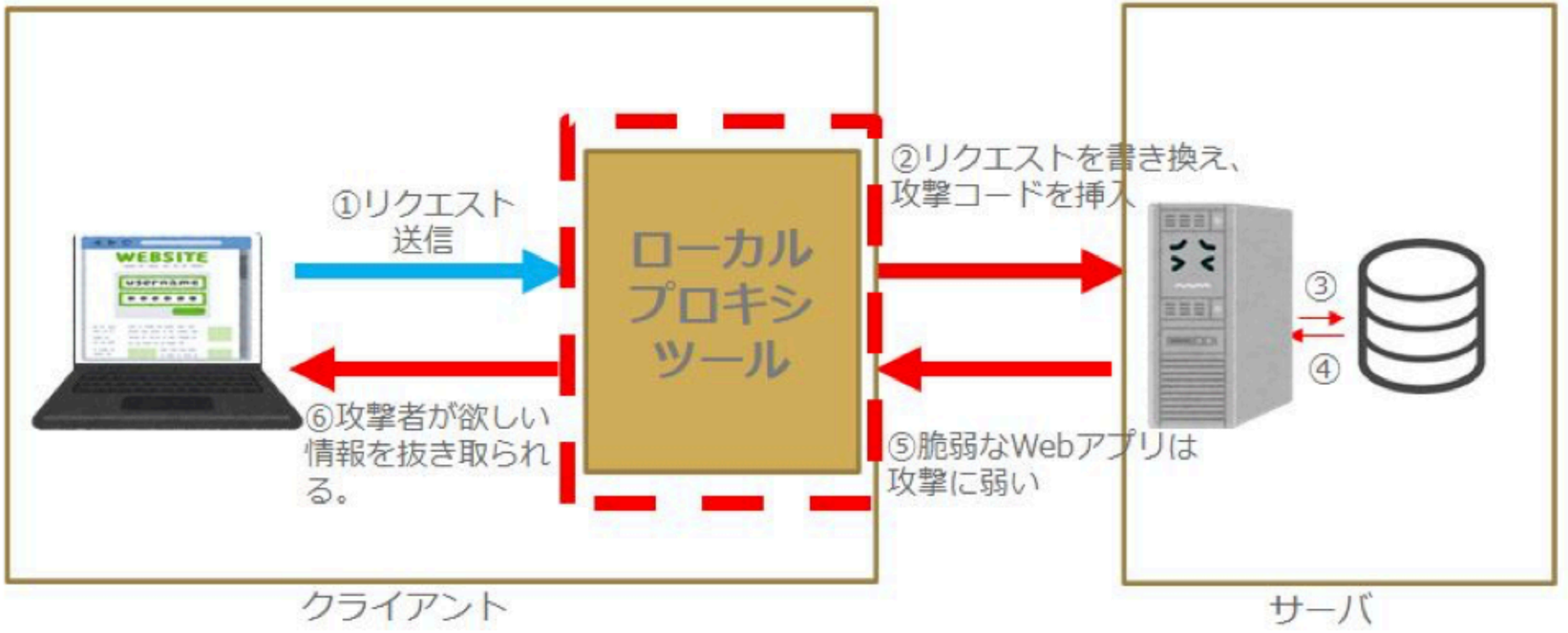
どうやってハッキングするのか

ここのリクエスト部分をいじって、結果をみて脆弱性を見つける



どうやってハッキングするのか

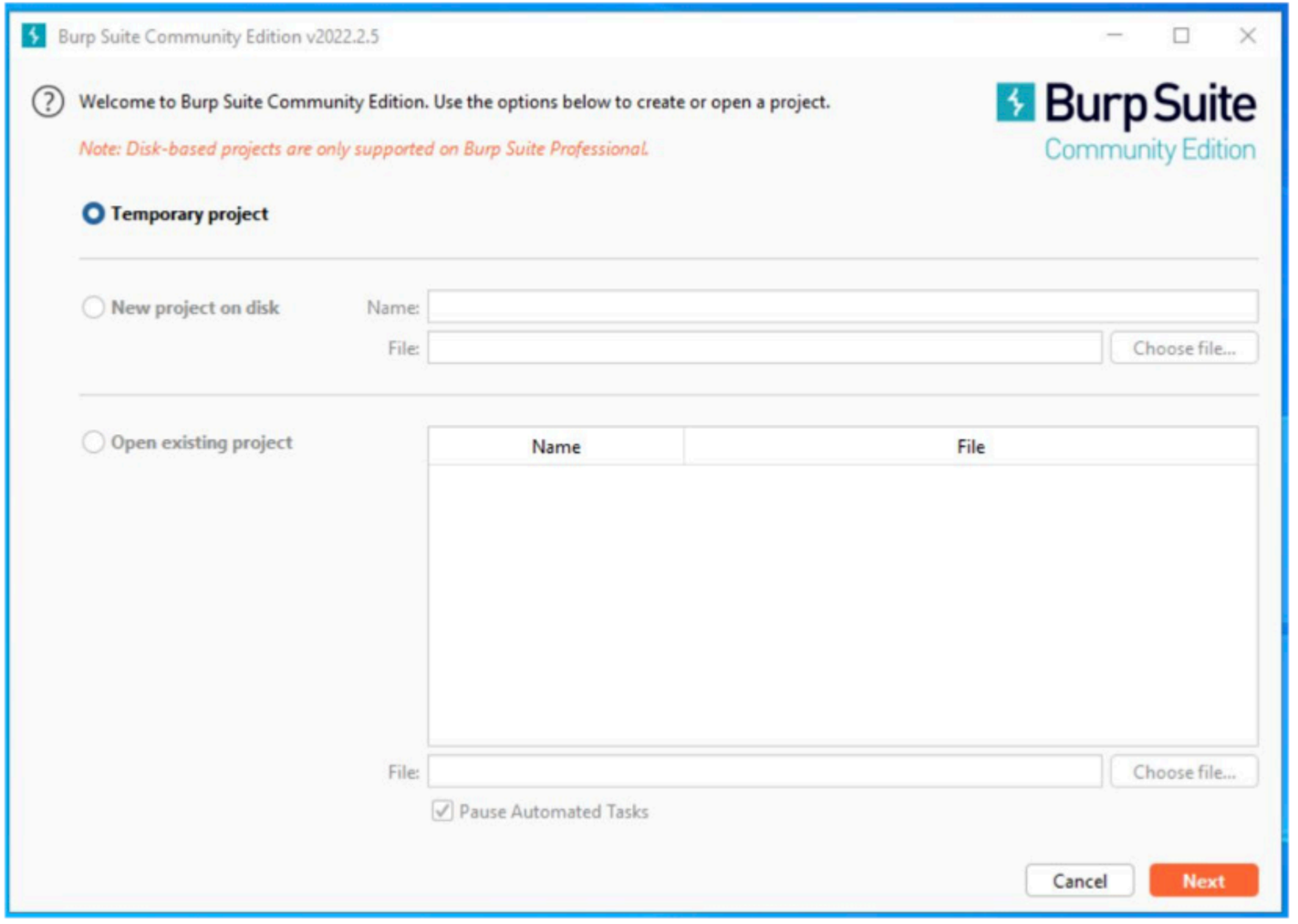
ローカルプロキシツールでWebアプリの脆弱性をみつける



今日使う「Burp Suite」もローカルプロキシツール！

Burp Suiteのセットアップ

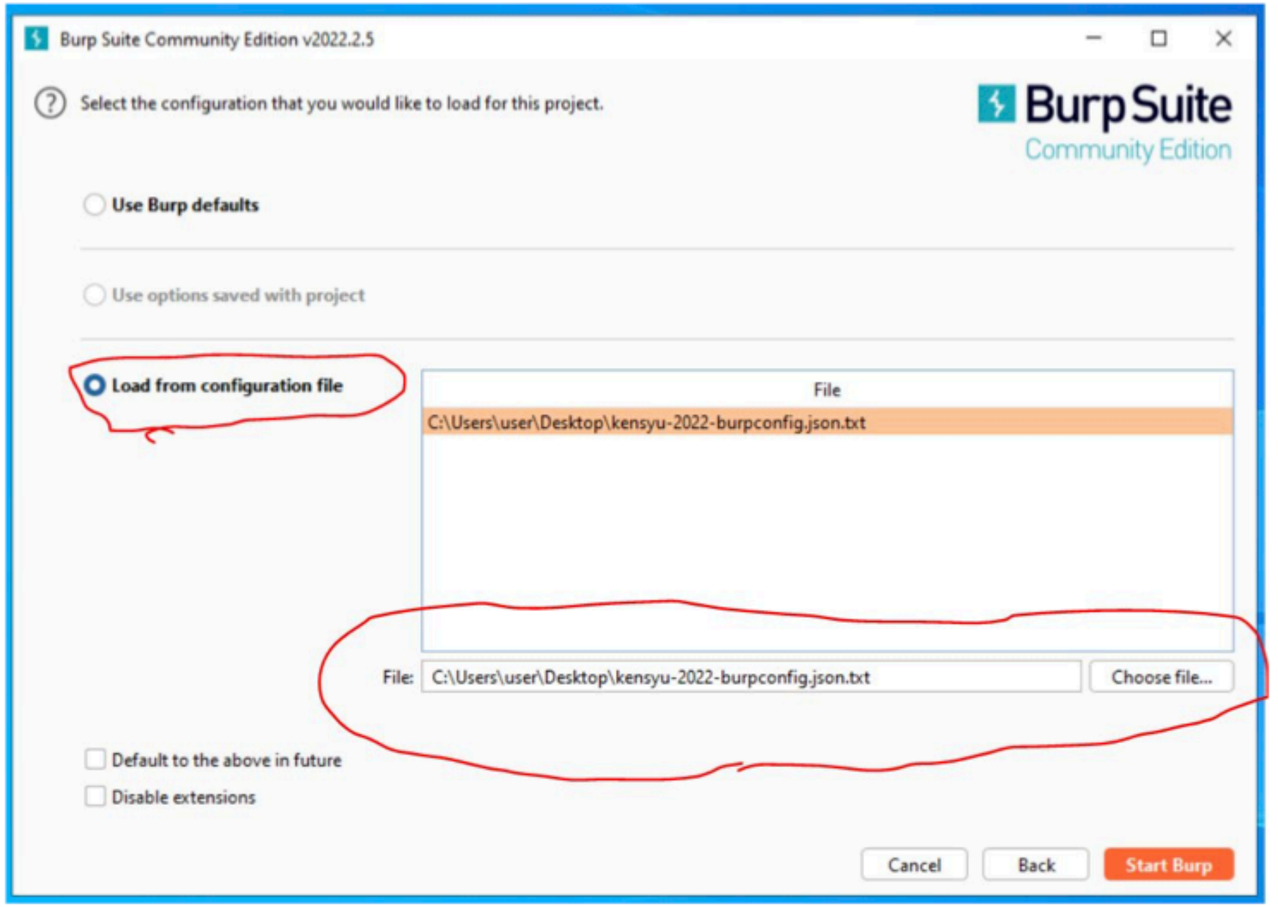
- Temporary projectを選択。



Burp Suiteのセットアップ

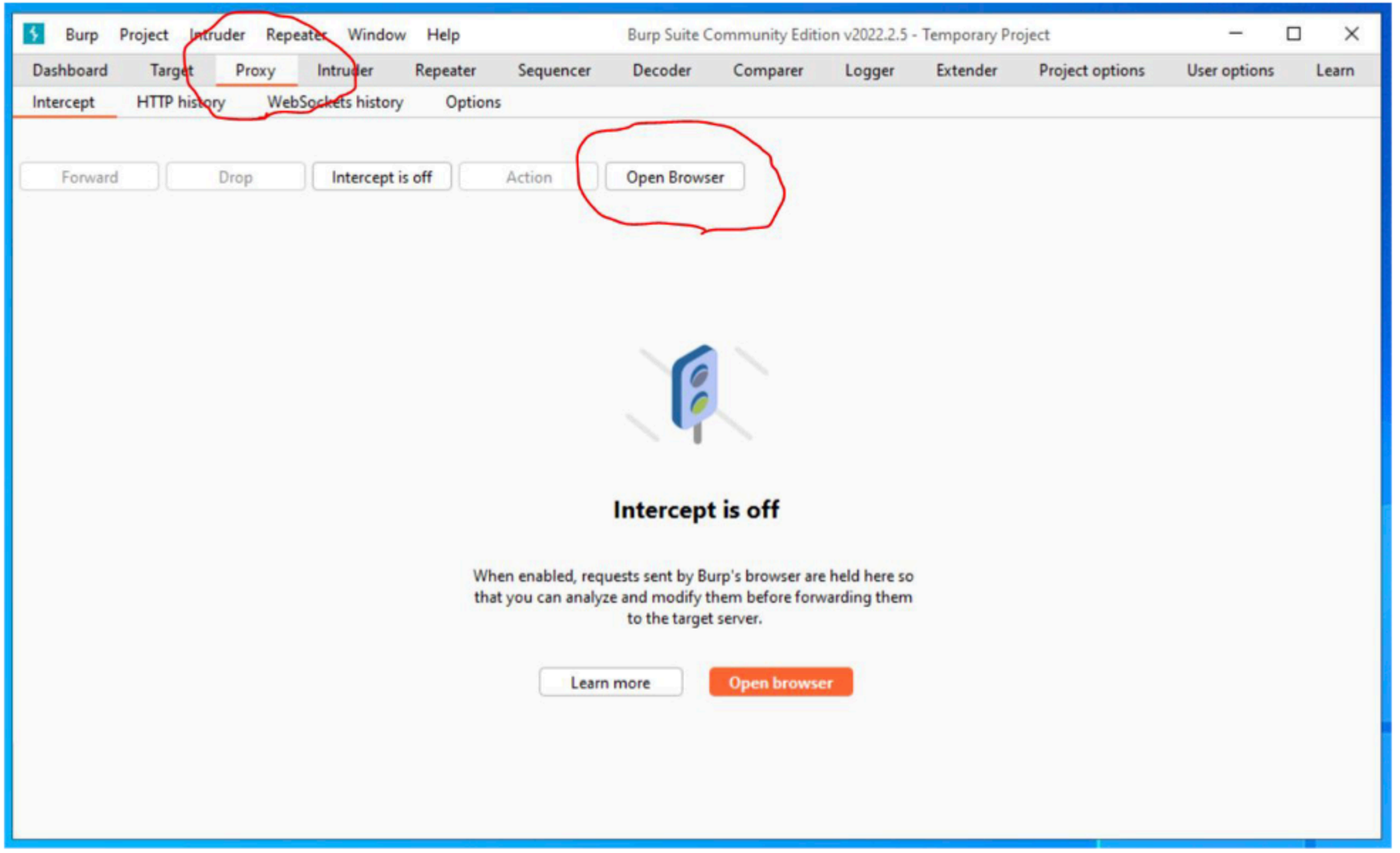
MIXI CONFIDENTIAL

- Load from configuration fileからコンフィグを読み込んでください。
  - 使用するコンフィグはこちら。これをローカルに落としてからLoadしよう。
    - Confidential



Burp Suiteのセットアップ

- 「Proxy」→「Open Browser」



Burp Suiteのセットアップ

MIXI
CONFIDENTIAL

- ブラウザが立ち上がります！
- 自分の環境のURL Confidential にアクセスしてみましょう

環境URL

Burp Suiteのセットアップ

MIXI
CONFIDENTIAL

- すると通信がせき止められます
- Proxyタブでせき止められた通信の内容を確認できます
  - 「intercept」ボタンがONになっていると、ブラウザから送られるリクエストはBurpがつかんで送信されないようになります
  - ONのままだとブラウジングできないのでクリックしてOFFにしましょう

Burp Suiteのセットアップ

MIXI
CONFIDENTIAL

- ログインを求められるのでご自身のGoogleアカウントでログインしてください。

Burp Suiteのセットアップ

MIXI
CONFIDENTIAL

- アクセスできたらOK！

おさえておきたいBurp Suiteの機能

MIXI
CONFIDENTIAL

基本的なもの

- Proxy>Intercept ー 通信のキャッチ
  - Intercept is on：通信をキャッチ
  - Intercept is off：通信をキャッチしない（通過）
  - Forward：キャッチした通信を送信
- Proxy>HTTP history ー 通信の履歴確認
  - Request：要求
  - Response：応答
- Repeater ー リクエストの再送
  - 挙動の確認に使う
  - 使い方
    - historyで再送したいリクエストに対しcontrol + タッチパッド
    - Send to Repeater選択
    - Repeater タブが光る！
    - Repeaterタブに移動
    - 「Send」で再送

Juice Shopを触ってみよう（通常動作確認）

MIXI
CONFIDENTIAL

1. 商品検索
    - キーワード「Apple」などで検索
2. アカウント登録
    - 任意のメアド「example01@exam.com」などで会員登録
3. ログイン
    - 作成済みアカウントでログイン
4. 商品購入
    - ログイン後＞ Basketに入れる＞ Checkout

BurpのHistoryを確認しながらサイト巡回してみよう。

おさえておきたいBurp Suiteの機能

MIXI
CONFIDENTIAL

ちょっと発展的なもの

- Intruder
  - 様々な試験値を試したいときに使う
  - 使い方
    i. 試験したいリクエストを「Send to Intruder」。
    ii. Intruder画面にて、試験したい箇所を「§」で囲む。

おさえておきたいBurp Suiteの機能

MIXI
CONFIDENTIAL

ちょっと発展的なもの

- Intruder
  - 様々な試験値を試したいときに使う
  - 使い方
    i. 試験したいリクエストを「Send to Intruder」。
    ii. Intruder画面にて、試験したい箇所を「§」で囲む。
    iii. 下記の試験値サンプルをコピーし、「Payloads」→「PayLoad Options」の「Paste」をクリック。
      - hoge
      - huga
      - piyo

おさえておきたいBurp Suiteの機能

MIXI
CONFIDENTIAL

ちょっと発展的なもの

- Intruder
  - 様々な試験値を試したいときに使う
  - 使い方
    i. 試験したいリクエストを
    「Send to Intruder」。
    ii. Intruder画面にて、試験したい箇所を
    「§」で囲む。
    iii. 下記の試験値サンプルをコピーし、
    「Payloads」→「PayLoad Options」の
    「Paste」をクリック。
      - hoge
      - huga
      - piyo
    iv. 「Start attack」をクリック。

おさえておきたいBurp Suiteの機能

MIXI
CONFIDENTIAL

ちょっと発展的なもの

- Intruder
  - 様々な試験値を試したいときに使う
  - 使い方
    i. 試験したいリクエストを
    「Send to Intruder」。
    ii. Intruder画面にて、試験したい箇所を
    「§」で囲む。
    iii. 下記の試験値サンプルをコピーし、
    「Payloads」→「PayLoad Options」の
    「Paste」をクリック。
      - hoge
      - huga
      - piyo
    iv. 「Start attack」をクリック。
    v. 「§」の箇所に試験値が送信される。
    その結果を一覧で確認できる。

OWASP Top10の脆弱性たち

MIXI
CONFIDENTIAL

API

- Broken Object Level Authorization
- Broken User Authentication
- Excessive Data Exposure
- Lack of Resources & Rate Limiting
- Broken Function Level Authorization
- Mass Assignment
- Security Misconfiguration
- Injection
- Improper Assets Management
- Insufficient Logging & Monitoring

Web

やや抽象化して解釈すると、
幾つかまとめられるものもある

- Broken Access Control
- Cryptographic Failures
- Injection
- Insecure Design
- Security Misconfiguration
- Vulnerable and Outdated Components
- Identification and authentication failures
- Software and Data Integrity Failures
- Security Logging and Monitoring Failures
- Server-Side Request Forgery (SSRF)

OWASP Top10の脆弱性たち

MIXI
CONFIDENTIAL

まずはこれを体感してみましょう

API

- Broken Object Level Authorization
- Broken User Authentication
- Excessive Data Exposure
- Lack of Resources & Rate Limiting
- Broken Function Level Authorization
- Mass Assignment
- Security Misconfiguration
- Injection
- Improper Assets Management
- Insufficient Logging & Monitoring

Web

- Broken Access Control
- Cryptographic Failures
- Injection
- Insecure Design
- Security Misconfiguration
- Vulnerable and Outdated Components
- Identification and authentication failures
- Software and Data Integrity Failures
- Security Logging and Monitoring Failures
- Server-Side Request Forgery (SSRF)

第一問

MIXI
CONFIDENTIAL

他人のカート(basket)を覗き見しよう

第一問

MIXI
CONFIDENTIAL

ヒント
自分のカートにアクセスした時のHTTP通信を見てみよう。
（/rest/basket/{0}）

第一問

MIXI
CONFIDENTIAL

他人のカート(basket)を覗き見しよう

答え
/rest/basket/6　の数字部分（カートID）を改変する。
例えばidに3を指定すれば、ユーザIDが3のユーザのカート情報が見られる

第一問

MIXI
CONFIDENTIAL

問題点

- アクセス制御が適切に行われていない

対策

- ユーザと紐づく情報を取得する際に、正当性検証を必ず行う
- そもそもカートIDが必要か？　も吟味する
  - セッションから参照でよいならその方がセキュア

第一問

MIXI
CONFIDENTIAL

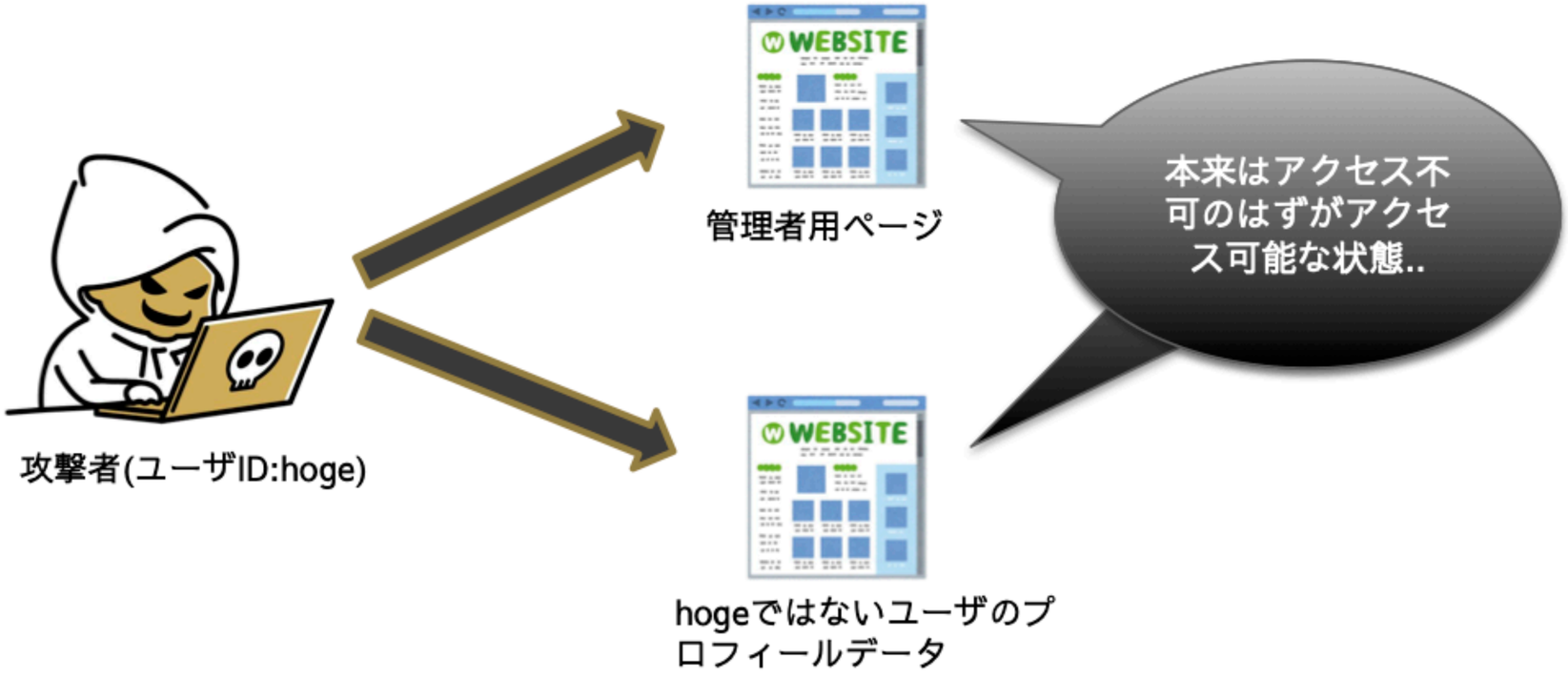
OWASP Top10では

- (API)Broken Object Level Authorization
- (API) Broken Function Level Authorization
- (Web) Broken Access Control

総じて、「アクセス制御を適切に行いましょう」という点が肝。

アクセス制御の不備とは？

アクセス(認証、認可、セッション管理など)の制御不備のこと。
本来その人がアクセスできないはずの情報にアクセスできてしまう問題。



アクセス制御の不備によって想定される被害

- 非公開情報の漏洩
- 権限外機能の操作

など（多岐にわたる

コード的な話

MIXI
CONFIDENTIAL

〈脆弱なコード例1（Ruby）〉

- 他ユーザの情報を見られちゃうコード(表示コンテンツの取得個所にて..)

```
user_id = requestobj.param[:user_id]  # リクエストのパラメータ値は改ざん可能！
userprofile = User.where(user_id).profile
view(userprofile)
```

ユーザから送信された値を使って

プロフィール情報を取得し

それを表示している！

- 攻撃者が他ユーザのuser_idを指定してリクエストを送信すると、
  プログラムはそのuser_idと紐づいた情報を表示してしまう

- 以下のようなことに気を付ける必要がある
  - リクエストパラメータにuser_idを含ませる必要はあるか？　セッションIDで十分ではないか？
  - 含ませるならその妥当性は検証しなくて大丈夫か？

コード的な話

MIXI
CONFIDENTIAL

〈脆弱なコード例2（Ruby）〉

- 管理者ページにアクセスできちゃうコード(セッションの検査個所にて..)

```
session = get_session(req.header[:cookie])  #「,admin: true」を入れていない！
if session.nil?
        raise “セッションがありません！”
end
do_adminsomefunction
```

- adminじゃなくてもif文に引っ掛からず処理が進行してしまう
- ログイン直後の画面でだけ権限確認をするのではなく、
  そこから辿れるリンク先の画面でも都度検証しましょう

対策（気をつけること）

- 権限管理をしっかりするためには、権限管理をしっかりしましょう（身もふたもないが..）
- 銀の弾丸は無い..
  - しっかり問題意識を持っておけば気づきやすくなるはず！

第二問

MIXI
CONFIDENTIAL

クロスサイトスクリプティング

- 「javascript:alert(`xss`)」このJavascriptを注入できる場所を探し、サイト上で動かしましょう！

クロスサイトスクリプティングとは

動的なページ生成時に攻撃者によってスクリプトが埋め込まれ、
被害者のブラウザ上でそのスクリプトが動作してしまう脆弱性。

参考

- (IPA) 安全なウェブサイトの作り方
  - https://www.ipa.go.jp/files/000017316.pdf

クロスサイトスクリプティングによって想定される被害

- なりすまし
- フィッシングサイトへの誘導
- など

勘所

- Webページが攻撃者によって**任意**に改ざんされるということ
  - 閲覧したユーザのCookie情報を攻撃者に送信するスクリプトを仕込む
  - 偽フォームを埋め込んで認証情報を攻撃者に送らせる

第二問

MIXI
CONFIDENTIAL

クロスサイトスクリプティング

- 「javascript:alert(`xss`)」←この Javascriptを注入できる場所を探し、サイト上で動かしましょう！

第二問

MIXI
CONFIDENTIAL

クロスサイトスクリプティング

- 「javascript:alert(`xss`)」←この Javascriptを注入できる場所を探し、サイト上で動かしましょう！

ヒント1

- クロスサイトスクリプティングは「入力値を使ってHTMLを構築する際に、外部から JavaScriptを注入できる」という問題でした。ということは、「入力値を使ってHTMLを構築している場所」が攻撃場所になりそう！？

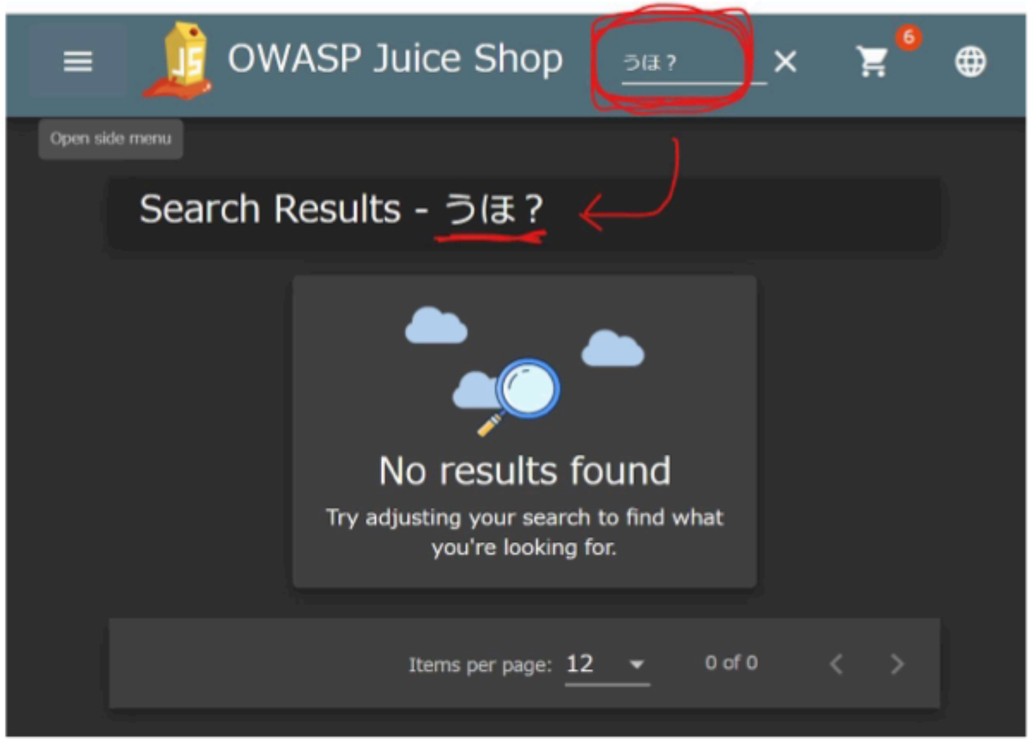
第二問

クロスサイトスクリプティング

- 「javascript:alert(`xss`)」←この Javascriptを注入できる場所を探し、サイト上で動かしましょう！

ヒント2

- Confidential /#/search　が怪しい…？



第二問

クロスサイトスクリプティング

- 「javascript:alert(`xss`)」←このJavascriptを注入できる場所を探し、サイト上で動かしましょう！

ヒント3

- 検索フォーム（Confidential /#/search）に以下を入力して結果を見てみましょう
  - 例1）<s>123</s>
  - 例2）<iframe src="https://mixi.co.jp">

第二問

MIXI
CONFIDENTIAL

クロスサイトスクリプティング

- 「javascript:alert(`xss`)」←この Javascriptを注入できる場所を探し、サイト上で動かしましょう！

答え

- 検索フォームにて、下記を入力する。
  - <iframe src="javascript:alert(`xss`)">

第二問

MIXI
CONFIDENTIAL

alert(`xss`)以外のスクリプトも動かしてみよう

- お試し1　(ログイン中のcookie情報取得)
  - <iframe src="javascript:alert(document.cookie)">
- お試し2　(他サイトへ移動させる)
  - <img src=1 onerror=window.location.href='https://mixi.co.jp'>

この応用で、攻撃者は好きなJavaScriptを動作させることができる…！

クロスサイトスクリプティングの対策

MIXI
CONFIDENTIAL

- 「<」,「>」,「"」,「'」など、ブラウザにとって意味を持つ文字（特殊文字）はエスケープ（文字列として解釈させる形式化）して出力する。
  - 今回のケースもこれが対策になる。
  - エスケープされた文字列の表示のされ方
    - Confidential
- 保険的対策として入力値検証やCSPヘッダの設定も有効。
- XSSは複数の種類があり、種類に応じて対策も異なる場合があるので注意
  - JavaScriptスキームを悪用するパターン
  - Dom構築時にスクリプトが仕込まれるパターン
- JavaScriptライブラリに問題がある場合は、使用しているライブラリを更新する。

動的にHTMLページを生成する際は、「レスポンスに攻撃者のスクリプトが埋め込める余地は無いか」というアンテナを張っておくこと！

コード的な話

MIXI
CONFIDENTIAL

〈脆弱なコード例〉

- ユーザが入力した検索ワードをレスポンスに表示する

```
keyword = requestobj.param[:keyword]  # keywordにScriptなどが入力されると..
responsehtml = “<body>” + keyword + “の検索結果一覧” + ”</body>”
```

ここに入力したScriptがそのまま埋め込まれる！

※例であって、Juice Shopのコードがこの通りになっているというわけでないです！

コード的な話

MIXI
CONFIDENTIAL

〈脆弱なコード例〉

- ユーザが入力した検索ワードをレスポンスに表示する

```
keyword = requestobj.param[:keyword]  # keywordにScriptなどが入力されると..
responsehtml = “<body>” + escape(keyword) + “の検索結果一覧” + ”</body>”
```

エスケープ用の関数を経由させて出力させよう

※例であって、Juice Shopのコードがこの通りになっているというわけでないです！

第二問

MIXI
CONFIDENTIAL

OWASP Top10では

- (API)Injection
- (Web)Injection

- 攻撃コードを「注入」し実行させること全般を指す。
  - SQL injection
  - Cross site scripting
  - OS command injection
  - etc

第三問

安く（不正に）購入してみよう

- 普通、商品の個数は1個以上しか買えない（当然）ものです。しかしJuice Shopでは実はマイナスの個数の注文ができます！　それを行ってみましょう。

第三問

安く（不正に）購入してみよう

ヒント

- カート内の数量変更時の通信を見てみよう。
  - Confidential /BasketItems/{n}

第三問

MIXI
CONFIDENTIAL

安く（不正に）購入してみよう

答え

- 数量にマイナスをつける
  - {"quantity":-2}

さすがにマイナスの値段で決済が進むとは考えにくいけれど、

100円の商品Aを -9個と、
1000円の商品Bを1個とを併せて買えば、
100円でBを1個買えるのは現実的かも？

おとく！

第三問

安く（不正に）購入してみよう

答え

- 数量にマイナスをつける
  - {"quantity":-2}

さすがにマイナスの値段で決済が進むとは考えにくいけれど、

100円の商品Aを -9個と、
1000円の商品Bを1個とを併せて買えば、
100円でBを1個買えるのは現実的かも？

おとく！

Juice Shopのような現物を取引するサービスならば発送段階などで攻撃に気づけるかもしれないが、電子マネーのチャージやゲーム内通貨など、現物やり取りが行われないサービスの場合、気づくこともできない可能性があるので注意！

例えばオーブでガチャをまわすとき、オーブ数量をマイナス指定するとオーブが増えたりする　かも.....
よくある！

第三問

MIXI
CONFIDENTIAL

問題点

- 入力値の検証が適切に行われていない

対策

- 入力値検証では「仕様的におかしくないか？」もチェックする
  - Syntax的におかしくないか？　は検証していても（[0-9]+じゃないと弾かれる等）、「仕様や意味を考えたときにおかしい値の検証」が漏れることもあるので注意。
  - 例えば生年月日なら、仮に自然数の入力であったとしても、2200年生まれと主張するユーザがいたらおかしくないか？　といった観点でもチェックしよう。

第三問

MIXI
CONFIDENTIAL

OWASP Top10では

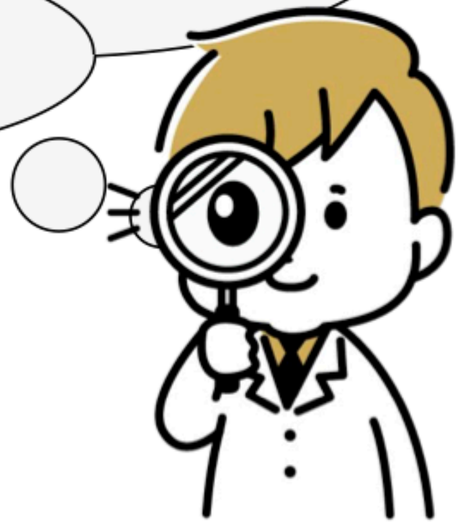
- (Web)Insecure Design

- 「設計上の欠陥」に起因する脆弱性全般を指す。
  - 安全な設計が行われていない
  - 暗号化すべき情報が暗号化されていない
  - ロジックの検証が不十分なため仕様外の操作が可能になっている

第四問

adminユーザとして会員登録しよう（権限昇格）

普通に会員登録すると、当然ながら作成したユーザはユーザ権限しか持ちません。
しかしうまく脆弱性を突くと、管理者権限を有するユーザを作成できるぞ！



第四問

MIXI
CONFIDENTIAL

adminユーザとして会員登録しよう（権限昇格）

ヒント1

- 会員登録フォーム（Confidential /api/Users）の通信を見てみよう。どこか怪しいところはない？

第四問

MIXI
CONFIDENTIAL

adminユーザとして会員登録しよう（権限昇格）

ヒント2

- レスポンスの「"role":"customer"」部分が怪しい！

第四問

MIXI
CONFIDENTIAL

adminユーザとして会員登録しよう（権限昇格）

答え

- 「,"role":"admin"」を無理やりjsonに付与した、下記のような値で会員登録する。

{"email":"Confidential","password":"hogehoge","passwordRepeat":"hogehoge","securityQuestion":{"id":4,"question":"Father's birth date? (MM/DD/YY)","createdAt":"2021-04-20T09:20:42.365Z","updatedAt":"2021-04-20T09:20:42.365Z"},"securityAnswer":"f","role":"admin"}

会員登録時のレスポンスのJsonを確認し、"role"がadminになっていればクリアー！

第四問

MIXI
CONFIDENTIAL

問題点

- role という追加パラメータの内容が解釈されて、ユーザ権限では本来実行できない会員作成が行われてしまった。

対策

- ユーザ権限によるリクエストの場合は、「"role":"customer"」部が変更され得ないようにする。
  - Juice Shop側（作り手側）としては「"role":"customer"」は本来の処理では送られてこないパラメータなので、対策観点から漏れやすい問題なのかもしれません。だからこそ注意をはらう必要があります。

第四問

MIXI
CONFIDENTIAL

OWASP Top10では

- (API)Mass Assignment

- 入力値のフィルタリングを行わずにデータにバインディングしてしまう問題のこと。
  - 今回の出題のように、「role」プロパティの値を指定する入力値を追加された際、それをフィルタリングせずに評価してしまい、結果、意図しない権限を付与してしまう　等

総合演習

攻撃者の気持ちになって、管理者アカウントに不正アクセスする流れを再現しよう！

総合演習

MIXI
CONFIDENTIAL

流れ

1. 管理者用ページを見つけよう
2. SQL Syntaxエラー（SQLITE_ERROR） を出してみよう
3. SQLインジェクションを悪用して、全ユーザの情報（Eメール、パスワードハッシュ）を抜き出そう
4. 「bender@juice-sh.op」ユーザでログインしよう
5. adminユーザでログインしよう

総合演習

1. 管理者用ページを見つけよう

総合演習

MIXI
CONFIDENTIAL

1. 管理者用ページを見つけよう

ヒント1

- Confidential #/search のHTMLソースを見てみよう。

総合演習

MIXI
CONFIDENTIAL

1. 管理者用ページを見つけよう

ヒント2

- Confidential #/search のHTMLソースを見てみよう。
- さらにmain.js内のpath部分を見てみよう。

総合演習

1. 管理者用ページを見つけよう

答え

- Confidential #/administration が管理者ページ
  - 開発者ツールで、main.js内を「administration」で検索すると見つかる。
  - アクセスしても403エラーになると思います。権限が足りないのでそれでOK。

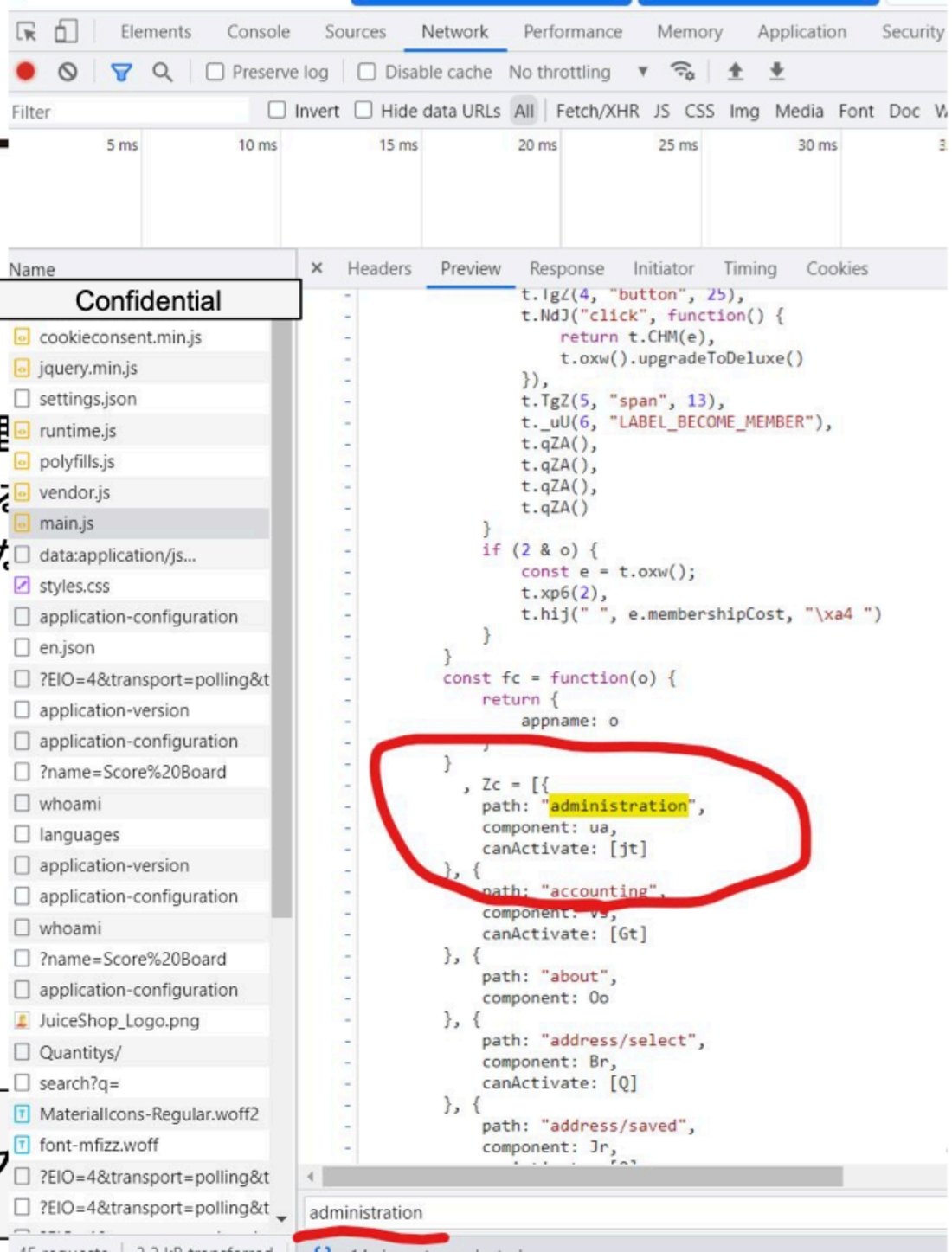
このページに不正アクセスできたら悪さできそうだな

総合演習

1. 管理者用ページを見つけよう

答え

- Confidential #/administration が管理
  - 開発者ツールで、main.js内を「administration」で検索す
  - アクセスしても403エラーになると思います。権限が足り



このページに不正アク

総合演習

2. SQL Syntaxエラー（SQLITE_ERROR） を出してみよう

不正アクセスのためにSQLインジェクションを狙う

SQLインジェクションとは？

MIXI
CONFIDENTIAL

アプリケーションが想定しないSQL文を実行させることにより、データベースシステムを不正に操作する攻撃方法のこと。また、その攻撃を可能とする脆弱性のこと。

参考

- (IPA) 安全なウェブサイトの作り方
  - https://www.ipa.go.jp/files/000017316.pdf

SQLインジェクションによって想定される被害

MIXI
CONFIDENTIAL

- 非公開情報の漏洩
- 情報の改ざん
- データの削除・破壊
- 認証回避による不正ログイン
- など

コード的な話

MIXI
CONFIDENTIAL

〈脆弱なコード例（Ruby）〉

```
username = requestobj.params[:name]

query = "INSERT INTO users (name, description) VALUES (username, \"hogehoge\")"

SQL.query(query)
```

- この例では、ユーザの入力値をそのままSQLクエリに使用しています。
- 「"」などを挿入されることにより、想定しないSQLクエリが生成されます。

以下のSQL文を入力した場合

- 「joe", "somebody"); DROP TABLE users; --」

作られるSQL文は以下のようになります。

```
INSERT INTO users (name, description) VALUES ("joe",
"somebody"); DROP TABLE users; --", "hogehoge");
```

上記のSQLインジェクションでusersテーブルが削除されてしまいます。。

SQLインジェクションの対策

MIXI
CONFIDENTIAL

プレースホルダという仕組みを使うことが対策となる。

```
INSERT INTO users (name, description) VALUES (username,
"hogehoge");
```

```
INSERT INTO users (name, description) VALUES (?, ?);
```

最初にSQL文の構造を決定しておく仕組みが「プレースホルダ」

ポイント

- SQLを準備する段階で SQL文の構文が確定し、後から SQL構文が変化することがないため安全。
  - 「 joe", "somebody"); DROP TABLE users; -- 」という入力がされた場合、description箇所が当該文字列になるだけになる。

プレースホルダ以外の対策

MIXI
CONFIDENTIAL

※プレースホルダが根本的対策。併せて実施しておくと良いという意味での紹介。

- 入力値チェック
  - 文字種、桁数、取り得る値のリスト等の範囲を仕様として絞り込める場合、もれなく入力値チェックを行うことは有効である
  - その際の入力値チェックはブラウザ上で動作するJavaScriptによるのではなく、Webサーバ側のアプリケーションプログラムによって行う必要がある。

- 特殊記号対策
  - シングルクォーテーション「'」のエスケープ
    - 例　「'」⇒「''」等

余談

MIXI
CONFIDENTIAL

おそらく実際に皆さんが書く処理はこんな感じ

- （Rubyの例）

```
username = requestobj.param[:name]


userprofile = User.where(username: username).profile


view(userprofile)
```

- あれ？　SQL文いなくない?
  - フレームワークによってSQLはコーディングから隠蔽されていることが多いかもしれない
  - 正直、モダンなフレームワークを使っていれば勝手に対策されているので、SQLインジェクションが作りこまれるケースは少ないかもしれない
  - 逆に言えばコーディングしているだけでは危険性を知れないかも、とも言えるので、今のうちに原理をちゃんと抑えておきましょう

総合演習

2. SQL Syntaxエラー（SQLITE_ERROR） を出してみよう

不正アクセスのためにSQLインジェクションを狙う

総合演習

2. SQL Syntaxエラー（SQLITE_ERROR） を出してみよう

ヒント1

- アプリケーションがSQLを発行していそうな機能を探し、構文を崩す。この辺。
  - Confidential /rest/user/login
  - Confidential /rest/product/search?q=Apple

不正アクセスのためにSQLインジェクションを狙う

総合演習

2. SQL Syntaxエラー（SQLITE_ERROR） を出してみよう

ヒント2

- ログイン時のSQL文（想像）
  - SELECT * FROM xxxx WHERE id = 'test@example' and password = 'password123';

- 検索時のSQL文（想像）
  - SELECT * FROM xxxx WHERE name like = '%Apple%';

不正アクセスのためにSQLインジェクションを狙う

総合演習

MIXI CONFIDENTIAL

2. SQL Syntaxエラー（SQLITE_ERROR） を出してみよう

答え

- SQL文に利用される送信パラメータの末尾等にシングルクォーテーション「'」等を入れることで、意図的にSQLシンタックスを壊す。
  - Confidential /rest/user/login の「email」の値を改ざん
    - SELECT * FROM xxxx WHERE id = 'test@example'' and password = 'password123';
  - Confidential /rest/product/search?q=Apple の「q」の値を改ざん
    - SELECT * FROM xxxx WHERE name like = '%Apple'%';

SQLインジェクションできる場所を見つけたぞ！

総合演習

3.　SQLインジェクションを悪用して、全ユーザの情報（Eメール、パスワードハッシュ）を抜き出そう

SQLインジェクションでいざ情報奪取！

総合演習

3. SQLインジェクションを悪用して、全ユーザの情報（Eメール、パスワードハッシュ）を抜き出そう

ヒント1

- 商品検索のリクエストを見てみよう。
  - Confidential /rest/product/search?q=apple

- 先ほどのログイン時のSQLITE_ERROR内容から、テーブル名、カラム名が判明している。つまりここで発行されるSQL文は下記のよう。

- "SELECT * FROM Users WHERE email = 'test001@example.com'' AND password = 'b0baee9d279d34fa1dfd71aadb908c3f' AND deletedAt IS NULL"

SQLインジェクションでいざ情報奪取！

総合演習

3. SQLインジェクションを悪用して、全ユーザの情報（Eメール、パスワードハッシュ）を抜き出そう

ヒント2

- UNIONという仕組みを使ってみよう。

UNIONとは

2つ以上のSELECT文の結果を統合する仕組みのこと。

name	kind	nakigoe
pochi	dog	wanwan
tama	cat	nyaaaaaan!!!

name	nickname	syumi
yamada	yamachaso	tozan
tanaka	nakata	nyaaaaaan!!!

SELECT name,kind,nakigoe FROM animals **UNION** SELECT name,nickname,syumi FROM friends;

UNION

name	kind	nakigoe
pochi	dog	wanwan
tama	cat	nyaaaaaan!!!
yamada	yamachaso	tozan
tanaka	nakata	nyaaaaaan!!!

商品テーブルとUsersテーブルを結合(UNION)してみよう！

※UNIONでつなぐデータの表は、カラム数と型を統一しないとエラーになるので注意。

総合演習

MIXI
CONFIDENTIAL

3. SQLインジェクションを悪用して、全ユーザの情報（Eメール、パスワードハッシュ）を抜き出そう

答え

- 商品検索機能に下記のようなリクエストを送信することで、SQLインジェクションにより商品テーブルとUsersテーブルの情報が結合され、ユーザ情報を取得できます。

- /rest/product/search?q=apple%27))%20UNION%20SELECT%20email,password,3,4,5,6,7,8,9%20FROM%20Users--
  - ※%27は「'」、%20は「 」をそれぞれURLエンコードした値

総合演習

MIXI CONFIDENTIAL

3. SQLインジェクションを悪用して、全ユーザの情報（Eメール、パスワードハッシュ）を抜き出そう

解説

まずはSQLエラーから収集できた情報を整理してみよう。

- 商品検索時のSQLエラーから収集できたProductsテーブルの情報
  - SELECT * FROM Products WHERE ((name LIKE '%APPLE'%' OR description LIKE '%APPLE'%') AND deletedAt IS NULL) ORDER BY name
    - テーブル名： Products
    - カラム名： name, description, deletedAt
- ログイン時のSQLエラーから収集できたUsersテーブルの情報
  - SELECT * FROM Users WHERE email = 'test1@example.com'' AND password = '79ac8e0c5fbb7fffa893660a9f581303'
    - テーブル名： Users
    - カラム名： email, password ← 欲しいのはコレ！

SQLインジェクションでいざ情報奪取！

総合演習

MIXI
CONFIDENTIAL

3. SQLインジェクションを悪用して、全ユーザの情報（Eメール、パスワードハッシュ）を抜き出そう

解説

次に、ProductsテーブルとUsersテーブルの結合を試してみると。。

- SELECT * FROM Products WHERE ((name LIKE '%APPLE%')) UNION SELECT email, password FROM Users--%' OR description LIKE '%Apple'%') AND deletedAt IS NULL) ORDER BY name

下記のエラーになるはずです。

- SQLITE_ERROR: SELECTs to the left and right of **UNION do not have the same number of result columns**
  - UNION時には、連結するカラム数を同じにする必要があります。

総合演習

MIXI CONFIDENTIAL

3.　SQLインジェクションを悪用して、全ユーザの情報（Eメール、パスワードハッシュ）を抜き出そう

解説

name	description	deletedAt	?	?	?	?	?	?
Apple Juice	The all-time classic.	null	?	?	?	?	?	?
Apple Pomace	hoge	null	?	?	?	?	?	?

SELECT * FROM Products WHERE ~~~

UNIONするためにはカラム数を統一する必要がある。
でもProductsのカラム数は不明。

email	password	3	4	5	6	7	8	9
hoge@juice.sh.op	3c2a~~	3	4	5	6	7	8	9
huga@juice.sh.op	963e~~	3	4	5	6	7	8	9

UNION　SELECT email,password,3,4,5,6,7,8,9 FROM Users--

カラム数を合わせるために、
ダミー値の3,4,5...を順番に足していく。
9まで足すとカラム数がそろい、UNIONが成功する。

SQLインジェクションでいざ情報奪取！

総合演習

MIXI
CONFIDENTIAL

3. SQLインジェクションを悪用して、全ユーザの情報（Eメール、パスワードハッシュ）を抜き出そう

解説

つまり、カラム数が合うまで下記のように試行を続けると、エラーが発生しなくなる時が訪れる。

それが答え。

1. /rest/product/search?q=apple%27))%20UNION%20SELECT%20email,password%20FROM%20Users--
2. /rest/product/search?q=apple%27))%20UNION%20SELECT%20email,password,3%20FROM%20Users--
3. /rest/product/search?q=apple%27))%20UNION%20SELECT%20email,password,3,4%20FROM%20Users--
4. /rest/product/search?q=apple%27))%20UNION%20SELECT%20email,password,3,4,5%20FROM%20Users--
5. /rest/product/search?q=apple%27))%20UNION%20SELECT%20email,password,3,4,5,6%20FROM%20Users--
6. /rest/product/search?q=apple%27))%20UNION%20SELECT%20email,password,3,4,5,6,7%20FROM%20Users--
7. /rest/product/search?q=apple%27))%20UNION%20SELECT%20email,password,3,4,5,6,7,8%20FROM%20Users--
8. /rest/product/search?q=apple%27))%20UNION%20SELECT%20email,password,3,4,5,6,7,8,9%20FROM%20Users--

SQLインジェクションでいざ情報奪取！

総合演習

MIXI
CONFIDENTIAL

3. SQLインジェクションを悪用して、全ユーザの情報（Eメール、パスワードハッシュ）を抜き出そう

答え（再掲）

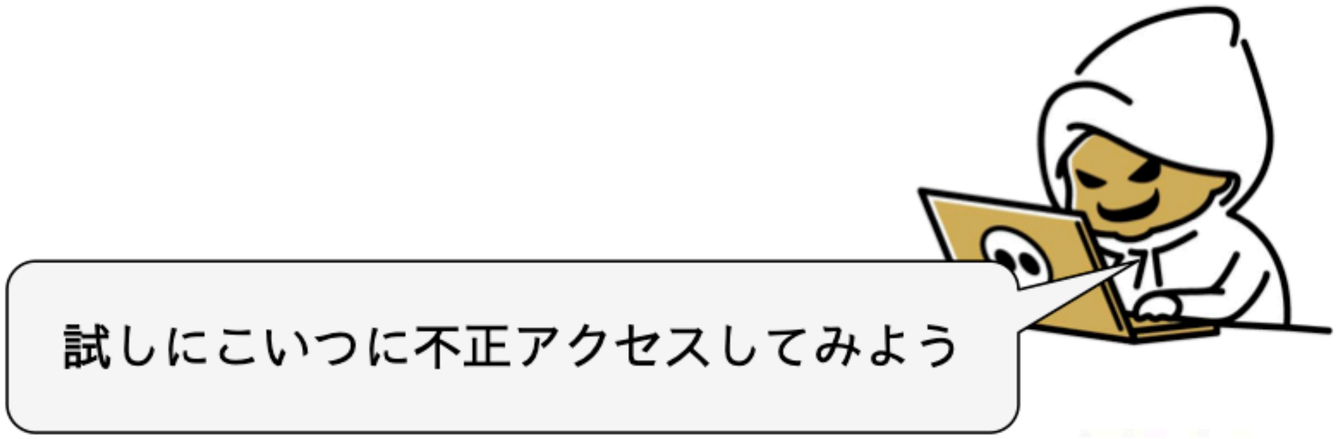
- 商品検索機能に下記のようなリクエストを送信することで、SQLインジェクションにより商品テーブルとUsersテーブルの情報が結合され、ユーザ情報を取得できます。

- /rest/product/search?q=apple%27))%20UNION%20SELECT%20email,password,3,4,5,6,7,8,9%20FROM%20Users--
  - ※%27は「'」、%20は「 」をそれぞれURLエンコードした値

ユーザー一覧の情報を奪取できた！

総合演習

4. 「bender@juice-sh.op」ユーザでログインしよう



総合演習

4. 「bender@juice-sh.op」ユーザでログインしよう

答え

- Emailの値を「bender@juice-sh.op'--」にする
  - SELECT * FROM Users WHERE email = 'bender@juice-sh.op'--' AND password = 'c4ca4238a0b923820dcc509a6f75849b' AND deletedAt IS NULL

総合演習

5. adminユーザでログインしよう

いざ、管理者ユーザに不正アクセスしてみよう！

総合演習

MIXI
CONFIDENTIAL

5. adminユーザでログインしよう

ちなみに、突破の仕方は2種類あります

- なお、パスワードは下記スプシに記載のもののどれかです。
  - Confidential

いざ、管理者ユーザに不正アクセスしてみよう！

総合演習

MIXI CONFIDENTIAL

5. adminユーザでログインしよう

答え1

- 先ほど取得したユーザ情報の一覧から、adminユーザのEメールは「admin@juice-sh.op」と分かる。これに対し、Eメールの値を「admin@juice-sh.op'--」とし、SQLインジェクションによるパスワード不要でのログインを行う。
  - SELECT * FROM Users WHERE email = 'admin@juice-sh.op'--' AND password = 'XXXXXXXXXXXXXX'

ログインできたら、冒頭で見つけた管理者ページ「Confidential /#/administration」にアクセスしてみよう。アクセスできるようになっているはず！

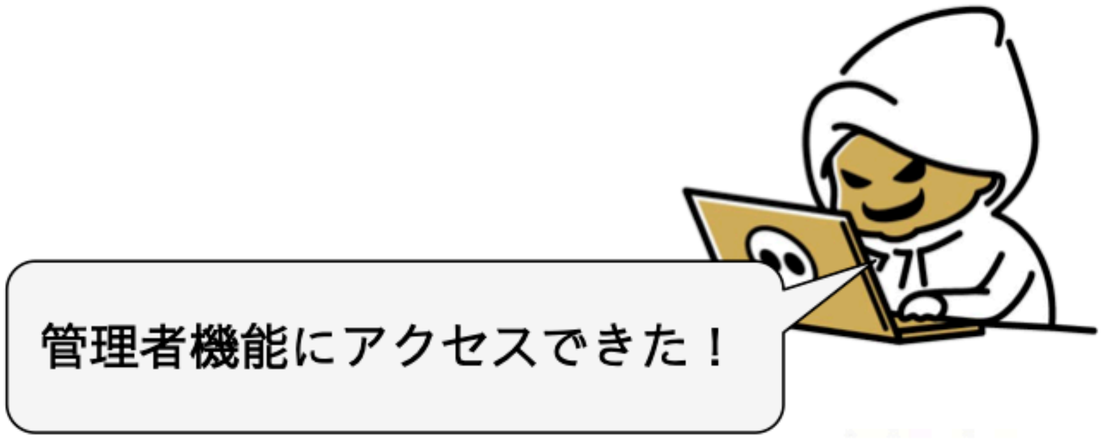
総合演習

MIXI
CONFIDENTIAL

5. adminユーザでログインしよう

答え2

- BurpSuiteの「Intruder」を利用し、総当たり攻撃を行うことでパスワードを割り出せる。
  - ※今回は簡略化のために300通りで値を総当たりしましたが、実際はもっと大量の値を総当たりしたり、既にどこかから漏洩したパスワードのリストを試行する可能性が高いです。



総合演習

MIXI
CONFIDENTIAL

OWASP Top 10での説明

- (API)Injection
- (Web)Injection
  - XSSのときに登場した問題。入力値からコードを注入できてしまう問題。
  - SQLインジェクションができてしまった点がこれに該当。

- (API)Broken User Authentication
- (Web)Identification and authentication failures
  - 認証機能をセキュアに実装できていない　という問題。
  - SQLインジェクションや総当たりをすることで不正に認証を突破できてしまった点がこれに該当。

- (API)Lack of Resources & Rate Limiting
  - リソースの要求サイズや数に制限が設けられていない　という問題。
  - ログイン試行が無数に行える(途中でロックアウトされない)点がこれに該当。

ハンズオンで紹介しきれなかったOWASP Top 10の脆弱性たち

MIXI
CONFIDENTIAL

API

- Broken Object Level Authorization
- Broken User Authentication
- Excessive Data Exposure
- Lack of Resources & Rate Limiting
- Broken Function Level Authorization
- Mass Assignment
- Security Misconfiguration
- Injection
- Improper Assets Management
- Insufficient Logging & Monitoring

Web

- Broken Access Control
- Cryptographic Failures
- Injection
- Insecure Design
- Security Misconfiguration
- Vulnerable and Outdated Components
- Identification and authentication failures
- Software and Data Integrity Failures
- Security Logging and Monitoring Failures
- Server-Side Request Forgery (SSRF)

(API/Web) Security Misconfiguration

MIXI
CONFIDENTIAL

不適切な設定をしてしまったことにより発生する問題全般

- 例
  - 古いバージョンのTLSが有効なまま
  - ライブラリ/フレームワーク等の設定値がセキュアでない
- この問題が招く脆弱性の一例として、XML External Entityという有名な脆弱性がある
  - 脆弱性対策演習で後日別途学習

(API/Web) Security Misconfiguration

MIXI
CONFIDENTIAL

想定される被害ケース例

- 幅が広いため多岐にわたるが、例えば設定不備に起因する脆弱性が突かれサーバ内部に置いていたファイルが漏洩する　等

(API/Web) Security Misconfiguration

MIXI
CONFIDENTIAL

対策

- これ！　というものは無い（使っているものによって着眼点は変わるため）が下記は意識を持っておくと良いと思います。
  - 開発やQA、本番環境それぞれで同じ設定になるようにする
    - 検証時はセキュアだったが、本番環境の設定では脆弱だったというリスクを減らす
  - 必要ない機能/コンポーネントは除いておく
    - コンポーネントの数だけ設定ミスで穴が開くリスクは増える

(API) Improper Assets Management

MIXI
CONFIDENTIAL

エンドポイントやドキュメントの管理が不適切になっている問題。

- 例
  - 古いAPIバージョンが残存したままになっている
  - デバッグ用APIが公開されている
  - 仕様をまとめたドキュメントが存在しない

(API) Improper Assets Management

MIXI
CONFIDENTIAL

想定される被害ケース例

- api.someservice.com/v2/userinfo というAPIがあったとして
- v2はセキュア、しかしv1には脆弱性がある状況で
- 攻撃者はapi.someservice.com/v1/userinfo 経由でユーザデータを不正取得できてしまった。

(API) Improper Assets Management

MIXI
CONFIDENTIAL

対策

- 公開するものは守る。守れていないものは公開しない。
  - /v1, /v2で例えると、/v1を残存させるならv1は守る。v1が脆弱なら公開を廃止する。
  - それをクリアに把握しておけるように、ドキュメント管理をしっかりしておきましょう。
    - このAPIは残っていていいのか？　等の判別のため。

(API) Insufficient Logging & Monitoring

MIXI
CONFIDENTIAL

ロギング/監視が不十分なため、
攻撃が行われた際に適切に対処することができない状況になっている問題。

- 例
  - ログが無いため、侵入者が何を行ったか分からない。影響範囲が特定できない。
  - 監視が無いため、侵入者の存在に気づけない。

(API) Insufficient Logging & Monitoring

MIXI
CONFIDENTIAL

想定される被害ケース例

- AWSアカウントにて、ある日コインマイニングを行っているEC2インスタンスを発見した。
- 原因を調べたところ、EC2を立てる権限を持ったユーザのアクセスキーが漏洩し、不正アクセスされたようだった。
- ただし、該当アクセスキーAPIログを残していなかったため、悪用被害範囲を特定することができなかった。

(API) Insufficient Logging & Monitoring

MIXI
CONFIDENTIAL

対策

- ログはできるだけ残しておく。
- 攻撃が行われた気づけるように監視をする。
  - AWS/GCPについてはセキュリティ室側の監視設定を施してあるアカウントであれば、一定の監視/ログ取得は行われています。
    - 攻撃が行われた場合はアラートが鳴る。
    - 管理系のログ（ユーザXがhogeというAPIをいつに実行した程度）は取得している。
    - 逆に言うと、エンドユーザのアクセスログ等は各自で取得しておくようにしましょう。
      - 認証失敗や不正行為の疑いのログなど。

(Web) Cryptographic Failures

MIXI
CONFIDENTIAL

暗号化に関連した問題の総称。

- 以前のverのTop 10では「機微な情報の露出」という名前で類似の問題が紹介されていた。

少々話がそれますが「機微な情報」とはなんぞや？　どう取り扱うべきか？　についてご紹介。
今回は以下を（簡単に！）取り上げてみます

- クレジットカード
- パスワード

クレカ情報を取り扱うときには

MIXI
CONFIDENTIAL

- **非保持化**しなさい！（原則）
- 保持するなら**PCIDSS**に準拠しなさい！

参考

- クレジットカード・セキュリティガイドライン
  - https://www.meti.go.jp/policy/economy/consumer/credit/2103creditsecurity.html

非保持化　について

MIXI
CONFIDENTIAL

非保持とは

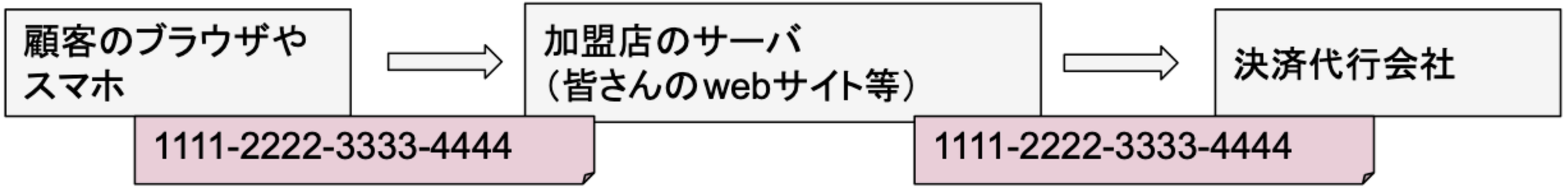
- 『保存』『処理』『通過』をすべて行っていない状態のこと。

- 保存
  - DBに保存。スプレッドシートに保存。etc,,
- 処理
  - コールセンターにて顧客のクレカ情報をPC入力。etc,,
- 通過
  - 顧客のPC→自社サーバ→決済代行会社サーバという流れでクレカ情報を送信。etc,,

非保持化　について

「非保持化？　保存さえしなきゃいいんでしょ？」

↑ダメ！！
　たとえ保存していなくても、例えば以下は『通過』にあたるのでＮＧ。



決済時は決済代行会社側のシステムに遷移させる等して、カード情報が自分たちのサーバを経由しないようにしましょう！

PCIDSS準拠　について

MIXI
CONFIDENTIAL

PCIDSSとは

- Payment Card Industry Data Security Standard
- クレジットカード会員データを安全に取り扱う事を目的として策定された、クレジットカード業界のセキュリティ**基準**のこと。
- 国際カードブランド5社(American Express、Discover、JCB、MasterCard、VISA)が共同で設立したPCI SSC(Payment Card Industry Security Standards Council)によって運用、管理されている。

「非保持化」をやらないなら、この基準を守れ！　ということ。

詳細は割愛ｍｍ

クレジットカード情報の流出で想定される被害

MIXI
CONFIDENTIAL

- カード利用者への不利益
  - 攻撃者に使われる
    - 本人認証が弱いプリペイドカードにチャージ
    - 偽造カードを作成し利用
  - 攻撃者に売られる
    - ブラックマーケットで換金
- 事業者への不利益
  - お客様へお詫びコスト
  - 不正に行われた買い物の損害賠償
  - カード決済の一時停止措置
  - コールセンター対応コスト
  - 再発防止コスト
  - **信頼を失う！**

パスワードを取り扱うときは

パスワードを保存する際、そのまま保存しないこと。

- 「データが盗まれてもパスワードは盗まれない」を目指す。
- ソルトと呼ばれる、ユーザごとに異なる文字列と連結した値を、ハッシュ化したうえで保存するのが良い。
  - 単にパスワードをハッシュ化しただけだと解析されてしまう可能性があるので注意。

参考

- (IPA) 安全なウェブサイトの作り方
  - https://www.ipa.go.jp/files/000017316.pdf

クレカとパスワードの話のまとめ

MIXI
CONFIDENTIAL

- クレジットカードの取扱い方
  - 非保持化する
  - しないならPCI DSSに準拠(監査なども含む)

- パスワードの取扱い方
  - 盗まれたとしても解析困難な形で保存する
    - 具体的には「ソルト付きハッシュ」という形式がセキュアな保存形式

(Web) Vulnerable and Outdated Components

MIXI
CONFIDENTIAL

脆弱性が残っているバージョンのコンポーネントを使用している問題。

(Web) Vulnerable and Outdated Components

MIXI
CONFIDENTIAL

想定される被害ケース例

- JVNDB-2021-005429
  - Log4jというJavaライブラリの脆弱性
  - 上記脆弱性が存在するバージョンのLog4jを使用していたために、攻撃者からインターネット経由で任意コード実行が行われる可能性

(Web) Vulnerable and Outdated Components

対策

- 脆弱性情報をウォッチしておく
  - CVE(Common Vulnerability and Exposures)やNVD(National Vulnerability Database)など
    - なお、脆弱性情報の見方については後ほどの章で別途説明します
- 不要なコンポーネントは取り除いておく

(Web) Software and Data Integrity Failures

MIXI
CONFIDENTIAL

悪意あるコードがコンポーネントに含まれている（含まれうる）問題。

- 外部ファイルを読み込む際、悪意あるものを指定されるかもしれない
- 読み込んでいる外部コードは改ざんされているかもしれない
- シリアライズデータが改ざんされて攻撃コードを注入されるかもしれない

想定される被害ケース例

- あるWebアプリでは、生成したオブジェクトをシリアライズしてCookieに保存し、そのオブジェクトが必要な際、Cookie値をデシリアライズして使用する仕様だった
- 攻撃者はCookie値を改ざんし、オブジェクトのデストラクタに攻撃コードを忍ばせたシリアライズデータをWebアプリに送信した
- すると、デストラクタ実行時、攻撃コードが実行された

(Web) Software and Data Integrity Failures

MIXI
CONFIDENTIAL

対策

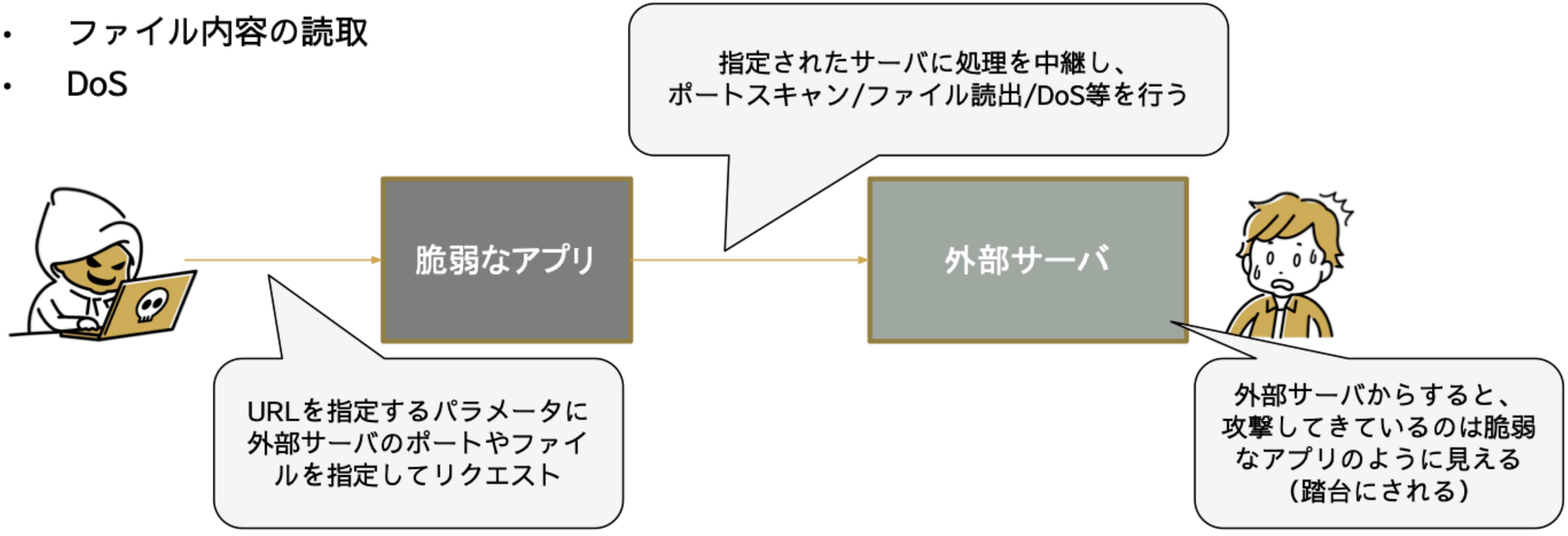
- 外部ファイルを読み込む際、悪意あるものを指定されるかもしれない
  - 外部からのファイルを読み込む際、想定していない外部ファイルを読み込まれない設計にする。例えばクライアントからURLを受け取り、そのURLのコードをそのまま読み込むといった構成を避ける。数値を受け取り、その数値と紐づいたファイルを読み込む等。
- 読み込んでいる外部コードは改ざんされているかもしれない
  - 信頼性の高い提供元の外部ライブラリを利用する。
- シリアライズデータが改ざんされて攻撃コードを注入されるかもしれない
  - そもそもシリアライズデータでオブジェクト等のコードを送受信する必要があるか見直す。
  - データの暗号化や署名付与を行い改ざんをできなくする。

(Web) Server-Side Request Forgery (SSRF)

アプリケーションを踏み台にして、内部/外部サーバにリクエストすることができる脆弱性。

想定される被害ケース

- ポートスキャン
- ファイル内容の読取
- DoS



(Web) Server-Side Request Forgery (SSRF)

MIXI
CONFIDENTIAL

対策

- ネットワーク層
  - SSRFが発生しうる機能をNW的に分離する
  - 想定している通信先以外への通信をFWでブロックする
- アプリケーション層
  - そもそも、リクエスト先URLをクライアントから直接受け取る構成をやめる。
    - 動的にリクエスト先を変えたい場合、固定値を受け取るようにする。例えば数値を受け取り、それに紐づいたURLにアクセスするなど。
  - 入力値検証を行う。
    - ホワイトリスト形式で検証する。
    - ブラックリスト形式や正規表現で検証すると、バイパスできる可能性が出てくるため極力行わない。

注意

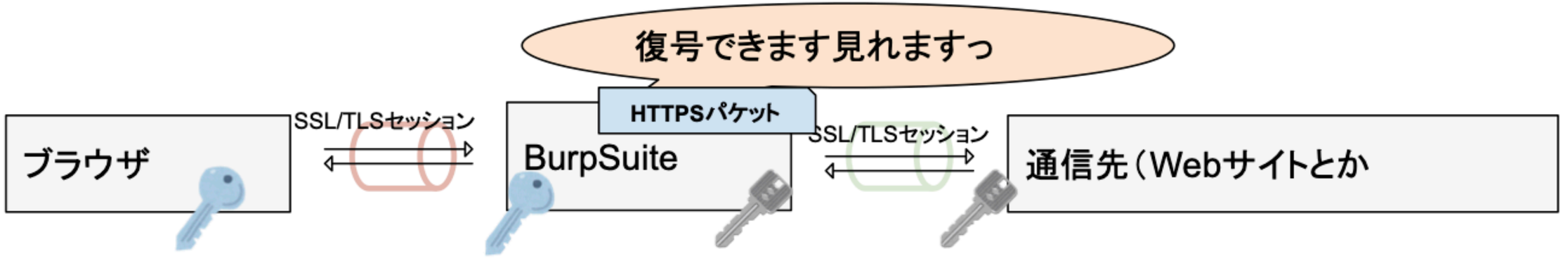
MIXI
CONFIDENTIAL

実運用中のWebサーバやWebアプリに対し、絶対にハッキングしないでください。
不正アクセス禁止法違反で法的責任に問われる事になりえます。

【余談】Burp Suiteってどんなツール？

MIXI
CONFIDENTIAL

- Burp SuiteってなんでHTTPS通信を見られてるの？
  - https って暗号化されてるんだから、経路上から見られんくない？
- 実はBurp SuiteはMan in the middleなツール
  - Burp Suiteはクライアントともサーバともssl/TLSをしている
- ブラウザからするとサーバと通信しているつもりがBurpと通信している
  - サーバからするとブラウザと通信しているつもりがBurpと通信している
- あれ、サーバ証明書は？
  - 当然、通常のブラウザでは（Firefoxとかで試すと）警告が出ます
  - ハンズオンではBurpの証明書が自動で信頼されている環境だっただけ
- フリーWiFi等の信頼できないNWでは攻撃者が同じことをしているかも..？



サーバ側を強固にするには？

MIXI
CONFIDENTIAL

- 公開するところ
  - **自分たちの作りこむもの（皆さんがコーディングする部分）**
    - **→ 脆弱性を知り、対策する**
  - 自分たちでは作りこまないもの（ライブラリやミドルウェア等）
    - → リスクを正しく評価し、必要に応じ[illegible]対策する
- 公開する必要のないところ
  - → アクセス制御を行って、アクセス自体を絞る

まずはここの話！

サーバ側を強固にするには？

MIXI
CONFIDENTIAL

- 公開するところ
  - 自分たちの作りこむもの（皆さんがコーディングする部分）
    - → 脆弱性を知り、対策する
  - **自分たちでは作りこまないもの（ライブラリやミドルウェア等）**
    - **→ リスクを正しく評価し、必要に応じ対策する**
- 公開する必要のないところ
  - → アクセス制御を行って、アクセス自体を絞る

次はここの話！

脆弱性対応時、調査のポイント

MIXI
CONFIDENTIAL

外部ライブラリ等、自分たちで作りこまないものの脆弱性対応は、

- 脆弱性があったら
- それを評価し
- 必要に応じそのモジュールをアップデートする

という流れになる。その際の評価ポイントは下記になる。

- どのソフトウェア？
- どのバージョン？
- どんな影響ある？
- どれくらい深刻なの？
- どう対策すればいい？

こういった情報の概要がまとまった脆弱性情報メディアに「CVE」がある。

CVEとは？

MIXI
CONFIDENTIAL

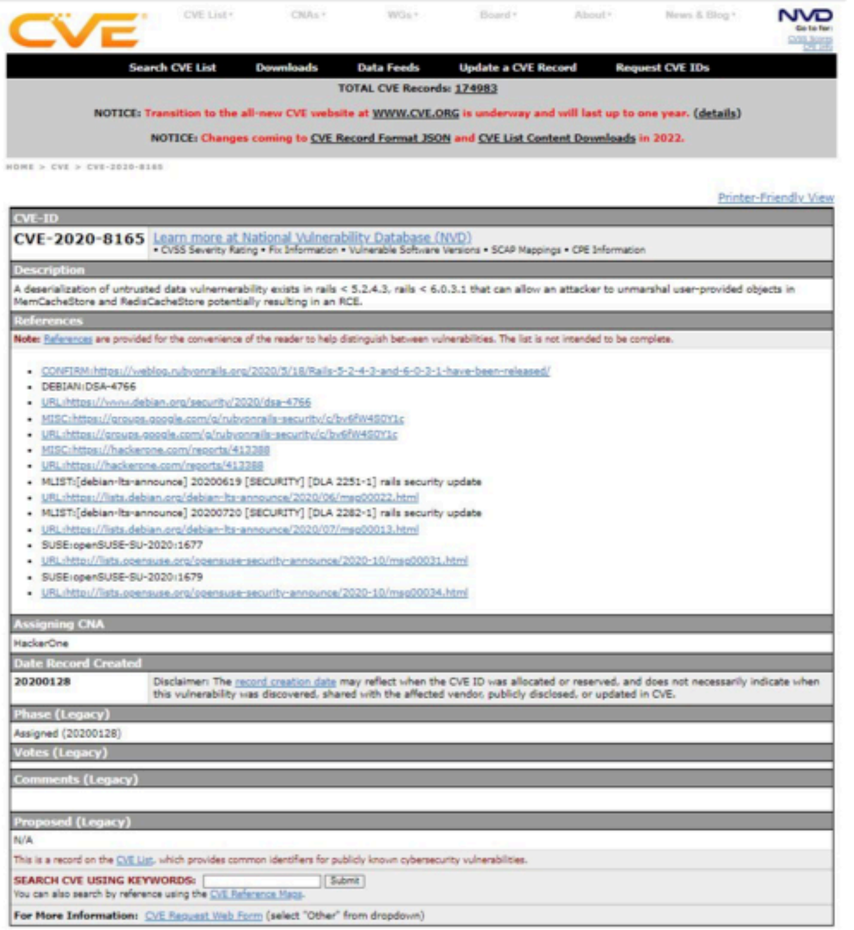
- Common Vulnerabilities and Exposuresの略。
- CVEとは、米MITRE（マイター）社が提供している、脆弱性を識別するための共通脆弱性識別子。
- 一般的にCVE番号、CVE IDと呼ばれている。

- 例：CVE-2020-8165
  - ※命名規則は、「CVE-YYYY- NNNN...N 」のようになっている。「YYYY」は発見された年数、NNNN...Nは一意の番号。

CVEの見方

- 例えば『CVE-2020-8165』なら、下記のようにCVEのサイトのname部分にIDを指定すれば確認できます。
  - https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-8165

- 見ると良い項目
  - CVE-ID：**NVD**へのリンクが記載されている。
  - Description：脆弱性概要が記載されている。
  - References：参考URLが記載されている。



CVE

Search CVE List | Downloads | Data Feeds | Update a CVE Record | Request CVE IDs

TOTAL CVE Records: 174983

NOTICE: Transition to the all-new CVE website at WWW.CVE.ORG is underway and will last up to one year. (details)

NOTICE: Changes coming to CVE Record Format JSON and CVE List Content Downloads in 2022.

Printer-Friendly View

CVE-ID

CVE-2020-8165 Learn more at National Vulnerability Database (NVD)

• CVSS Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings • CPE Information

Description

A deserialization of untrusted data vulnerability exists in rails < 5.2.4.3, rails < 6.0.3.1 that can allow an attacker to unmarshal user-provided objects in MemCacheStore and RedisCacheStore potentially resulting in an RCE.

References

Note: References are provided for the convenience of the reader to help distinguish between vulnerabilities. The list is not intended to be complete.

- CONFIRM:https://weblog.rubyonrails.org/2020/5/18/Rails-5-2-4-3-and-6-0-3-1-have-been-released/
- DEBIAN:DSA-4766
- URL:https://www.debian.org/security/2020/dsa-4766
- MISC:https://groups.google.com/g/rubyonrails-security/c/bv6fW4S0Y1c
- URL:https://groups.google.com/g/rubyonrails-security/c/bv6fW4S0Y1c
- MISC:https://hackerone.com/reports/413388
- URL:https://hackerone.com/reports/413388
- MLIST:[debian-lts-announce] 20200619 [SECURITY] [DLA 2251-1] rails security update
- URL:https://lists.debian.org/debian-lts-announce/2020/06/msg00022.html
- MLIST:[debian-lts-announce] 20200720 [SECURITY] [DLA 2282-1] rails security update
- URL:https://lists.debian.org/debian-lts-announce/2020/07/msg00013.html
- SUSE:openSUSE-SU-2020:1677
- URL:http://lists.opensuse.org/opensuse-security-announce/2020-10/msg00031.html
- SUSE:openSUSE-SU-2020:1679
- URL:http://lists.opensuse.org/opensuse-security-announce/2020-10/msg00034.html

Assigning CNA

HackerOne

Date Record Created

20200128 Disclaimer: The record creation date may reflect when the CVE ID was allocated or reserved, and does not necessarily indicate when this vulnerability was discovered, shared with the affected vendor, publicly disclosed, or updated in CVE.

Phase (Legacy)

Assigned (20200128)

Votes (Legacy)

Comments (Legacy)

Proposed (Legacy)

N/A

This is a record on the CVE List, which provides common identifiers for publicly known cybersecurity vulnerabilities.

SEARCH CVE USING KEYWORDS: Submit

You can also search by reference using the CVE Reference Maps.

For More Information: CVE Request Web Form (select "Other" from dropdown)

引用元：https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-8165,(2022/4/29)

詳細な脆弱性情報をみるにはNVD

MIXI
CONFIDENTIAL

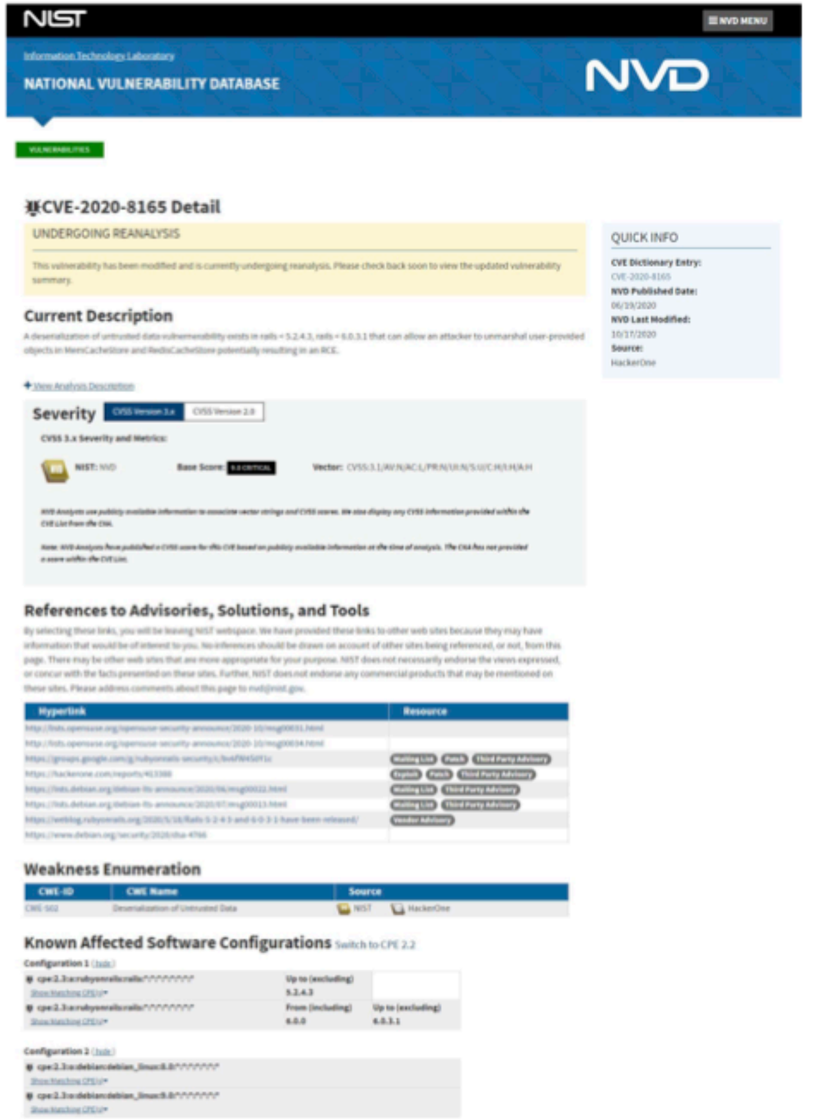
- National Vulnerability Database
- NIST(アメリカ国立標準技術研究所)が管理している、脆弱性情報データベースのこと。
- CVEで命名された脆弱性情報の詳細情報をNVDで提供している。

NVDの見方

CVE-2020-8165の例

- URL：https://nvd.nist.gov/vuln/detail/CVE-2020-8165

- 見ると良い項目
  - Current Description
    - 脆弱性の概要説明。
  - Severity
    - 「CVSS」の情報。脆弱性の深刻度。
  - References to Advisories, Solutions, and Tools
    - 関連する情報へのリンク。
  - Known Affected Software Configurations
    - 影響を受けるソフトウェアとバージョン。



引用元:https://nvd.nist.gov/vuln/detail/CVE-2020-8165, (2022/4/30)

CVSSとは

- Common Vulnerability Scoring System
- 脆弱性の深刻度をスコア（数値）で表すシステム。
  - 計算式に則り、脆弱性を0～10.0までの値で評価。
  - 10.0が最も深刻。
- NVDにはV2、V3の2バージョンが記載されている。

CVSS (V3) の見方

MIXI
CONFIDENTIAL

CVSSはVectorと呼ばれる下記の要素から算出されます。

- AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

例

- AV：N　ー>　攻撃元区分：ネットワーク
- AC：L　ー>　攻撃条件の複雑さ：低
- PR：N　ー>　必要な特権レベル：不要
- UI：N　ー>　ユーザ関与レベル：不要
- S：U　ー>　スコープ： 変更なし
- C：H　ー>　機密性への影響：高
- I：H　ー>　完全性への影響：高
- A：H　ー>　可用性への影響：高

このScoreは9.8 (Critial)！

脆弱性情報 (NVD/CVSS) の見方の勘所

MIXI
CONFIDENTIAL

脆弱性情報をどう読んだらいいか分からない場合、ひとまず以下の観点を確認すると良いと思います。

- 自分たちが対象製品、バージョンを使っていないか確認する。
- 攻撃元区分から攻撃者像を想定する。
- その攻撃者ができることを想定する。

例

- 自分たちが対象製品、バージョンを使っていないか確認する。
  - NVDのKnown Affected Software Configurations欄を確認したところ、自分たちのプロダクトで該当ソフトウェアの該当バージョンを使っていたことが分かった。
- 攻撃元区分から攻撃者像を想定する。
  - CVSSの脆弱性の攻撃元区分(AV)を確認したところ、「N」、つまりインターネット経由で攻撃可能と分かった。
- その攻撃者ができることを想定する。
  - NVDのCurrent Descripn欄を確認したところ、任意コード実行ができることが分かった。

以上を確認すると、「ヤバそう」かどうかのイメージが付きやすくなると思います。

日本の脆弱性情報収集サイト

MIXI
CONFIDENTIAL

- JVN
  - Japan Vulnerability Notes
  - 日本で使用されているソフトウェアなどの脆弱性関連情報とその対策情報を提供
  - https://jvn.jp/

- JVN iPedia
  - 国内外問わず日々公開される脆弱性対策情報のデータベース
  - https://jvndb.jvn.jp/

サーバ側を強固にするには？

- 公開するところ
  - 自分たちの作りこむもの（皆さんがコーディングする部分）
    - → 脆弱性を知り、対策する
  - **自分たちでは作りこまないもの（ライブラリやミドルウェア等）**
    - **→ リスクを正しく評価し、必要に応じ対策する**
- 公開する必要のないところ
  - → アクセス制御を行って、アクセス自体を絞る

次はここの話！

サーバ側を強固にするには？

MIXI
CONFIDENTIAL

- 公開するところ
  - 自分たちの作りこむもの（皆さんがコーディングする部分）
    - → 脆弱性を知り、対策する
  - 自分たちでは作りこまないもの（ライブラリやミドルウェア等）
    - → リスクを正しく評価し、必要に応じ対策する
- **公開する必要のないところ**
  - **→ アクセス制御を行って、アクセス自体を絞る**

続いてここの話！

【演習】Juice Shopをアクセス制御で守ろう

MIXI
CONFIDENTIAL

抑えておきたい「アクセス制御」の二つの思想

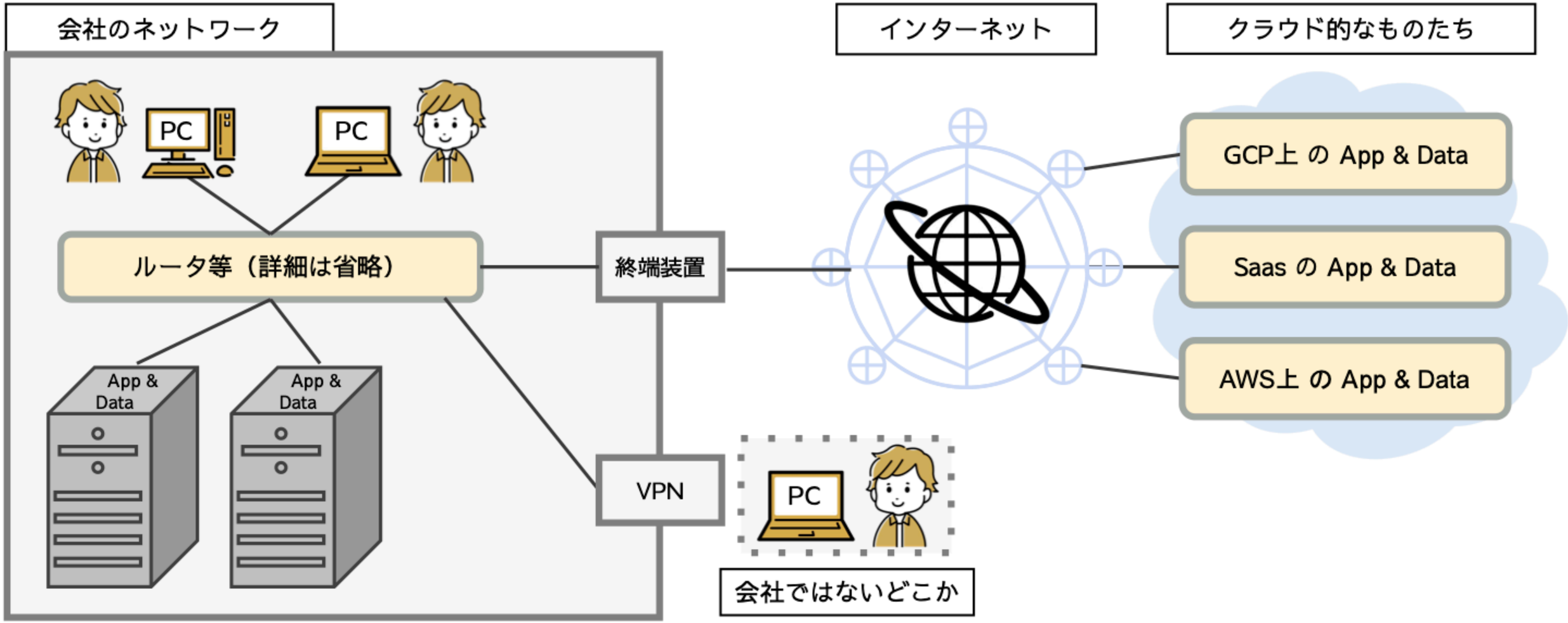
- 境界防御 → NWの境界線で守っていく考え方。
- ゼロトラスト → 弊社で取り組んでいるイマドキ(?)の考え方/やり方。

GCPの機能を使って、
境界防御とゼロトラストの思想それぞれでJuice Shopをアクセス制御してみましょう！

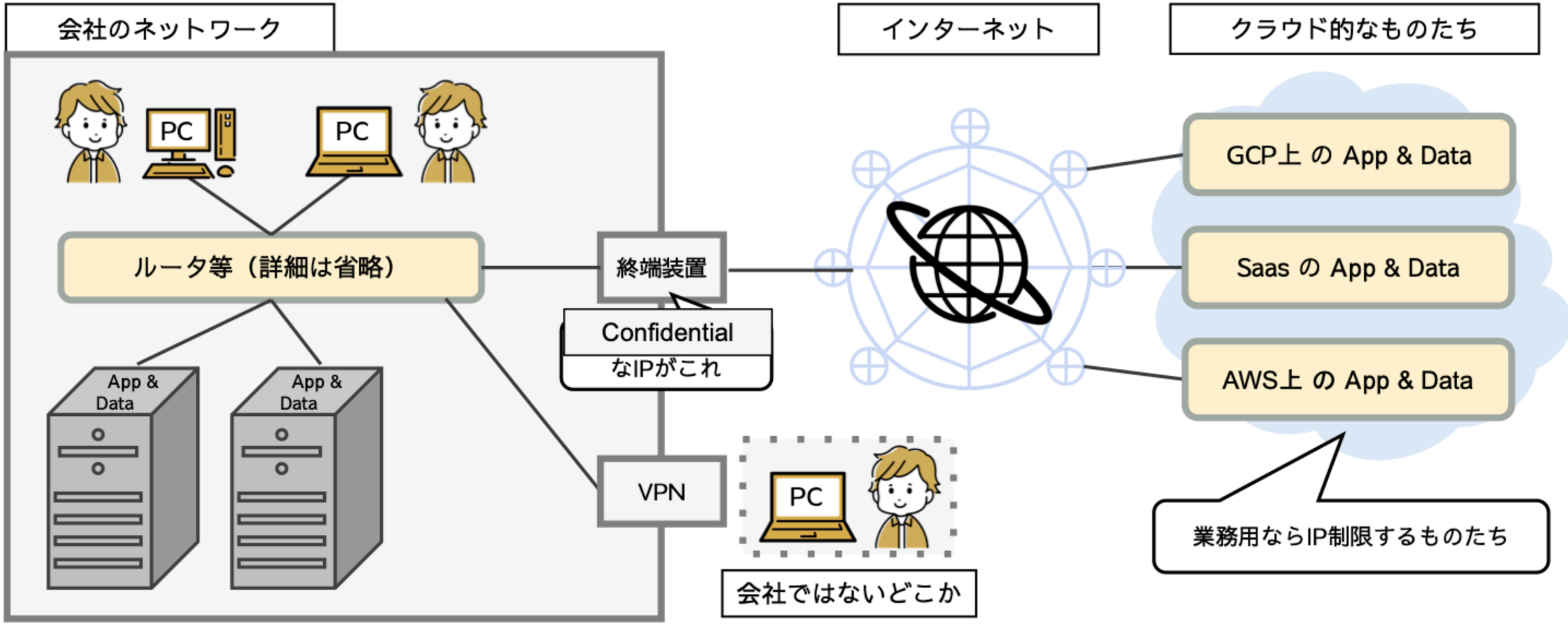
- Juice Shopは皆さんが運営しているサービスの試験環境だと想定してください。

境界防御とは

一般的な会社のネットワーク構成はこんな感じだと思います。

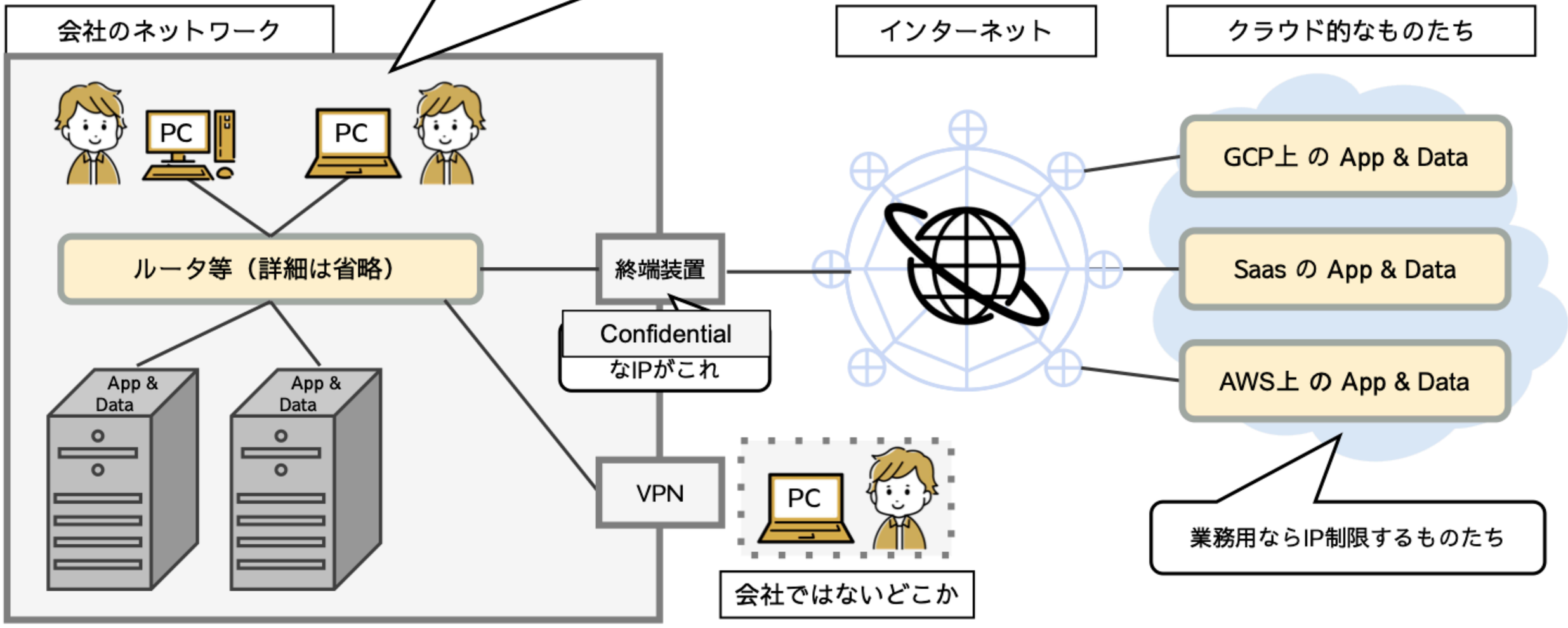


境界防御とは


会社のネットワーク
インターネット
クラウド的なものたち
PC
PC
ルータ等（詳細は省略）
終端装置
Confidential
なIPがこれ
App &
Data
App &
Data
VPN
PC
会社ではないどこか
GCP上 の App & Data
Saas の App & Data
AWS上 の App & Data
業務用ならIP制限するものたち

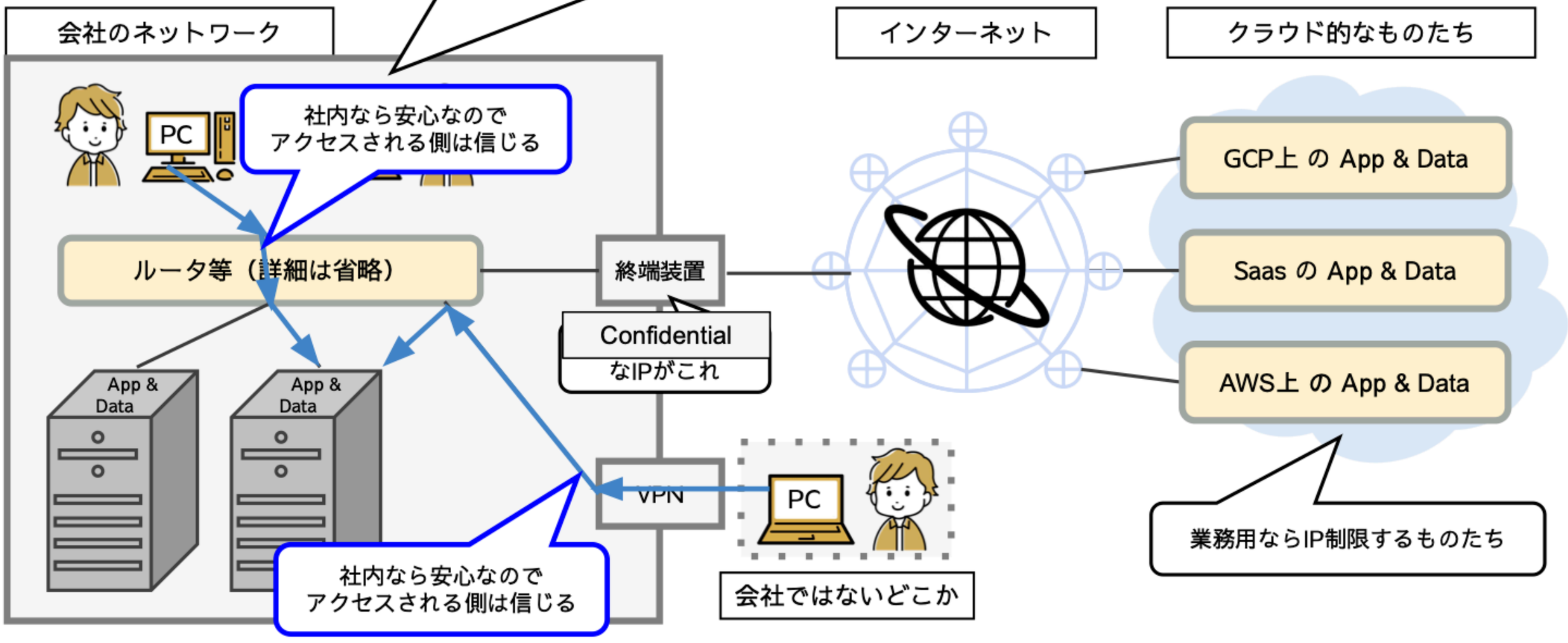
ここに繋ぐと認証等の各種対策/制約により下記が得られていた。

- 見知らぬ者が社内NWに居ない。→（盗聴者や攻撃者の排除）
- 攻撃監視やロギングが一定される。→（攻撃の検知や調査）
- 会社の終端装置から外に出る。→（IP制限による攻撃者排除）



ここに繋ぐと認証等の各種対策/制約により下記が得られていた。

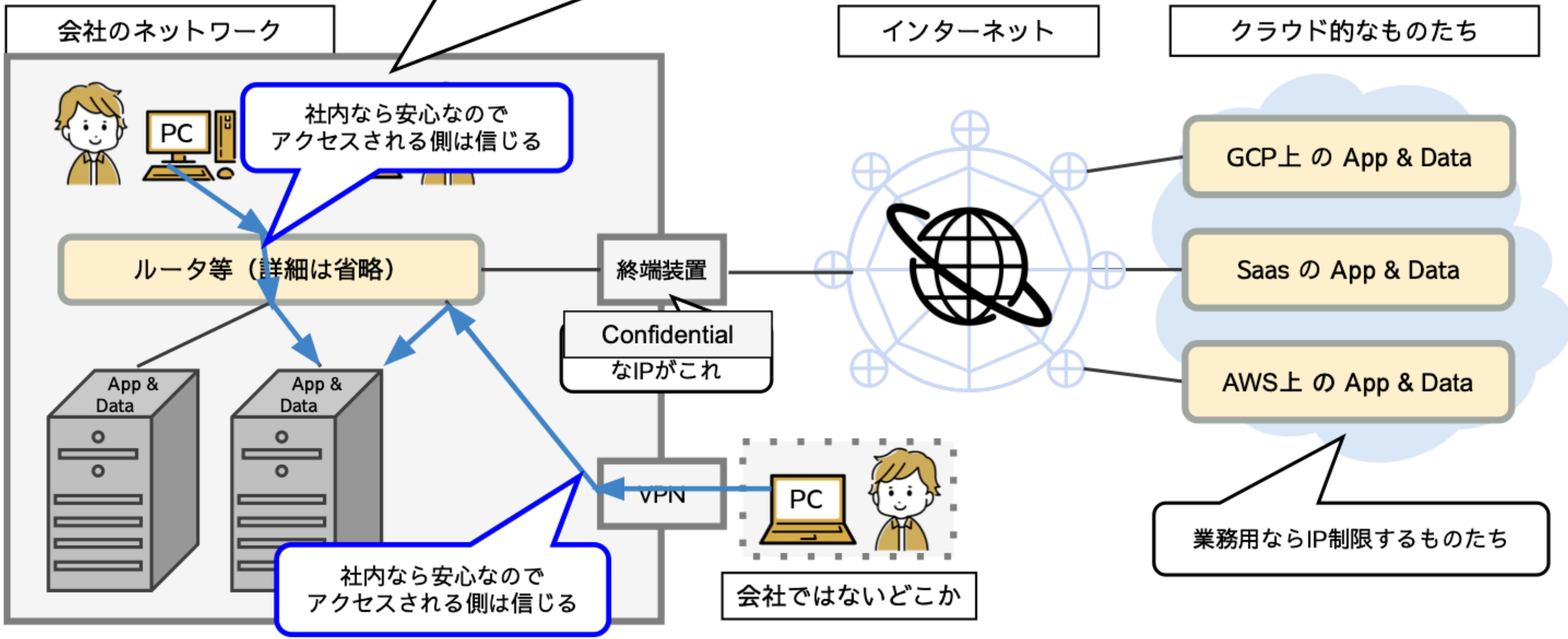
- 見知らぬ者が社内NWに居ない。→（盗聴者や攻撃者の排除）
- 攻撃監視やロギングが一定される。→（攻撃の検知や調査）
- 会社の終端装置から外に出る。→（IP制限による攻撃者排除）



ここに繋ぐと認証等の各種対策/制約により下記が得られていた。

- 見知らぬ者が社内NWに居ない。→（盗聴者や攻撃者の排除）
- 攻撃監視やロギングが一定される。→（攻撃の検知や調査）
- 会社の終端装置から外に出る。→（IP制限による攻撃者排除）

社内ネットワークなら安心という世界観



【演習】Juice Shopを境界防御にもとづいて守ろう

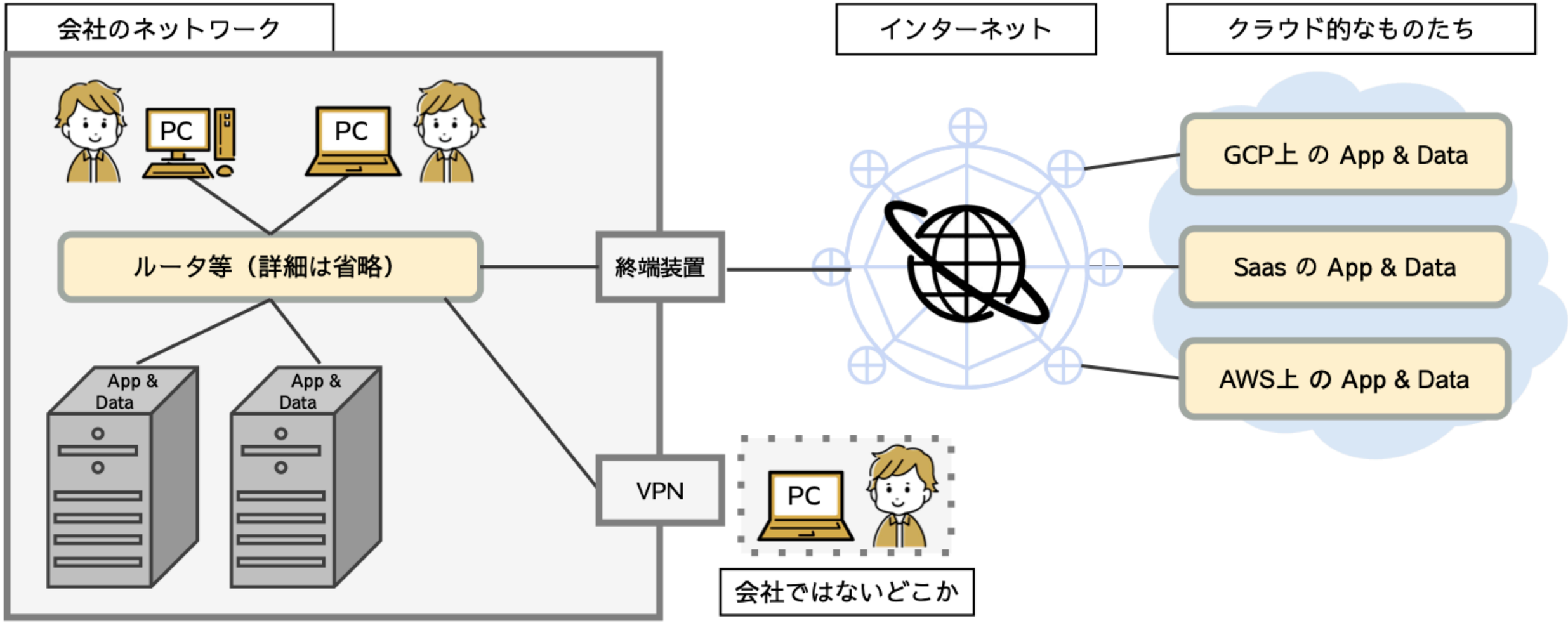
MIXI
CONFIDENTIAL

自身のJuice Shop環境にIP制限を施し、境界防御の考え方でのアクセス制御を行おう。

やること

- 下記手順書に則って進めてみましょう。
  - Confidential
  - 手順3の反映には時間が掛かるため、手順4は後から確認します。手順3までできた段階でいったん教えてください。

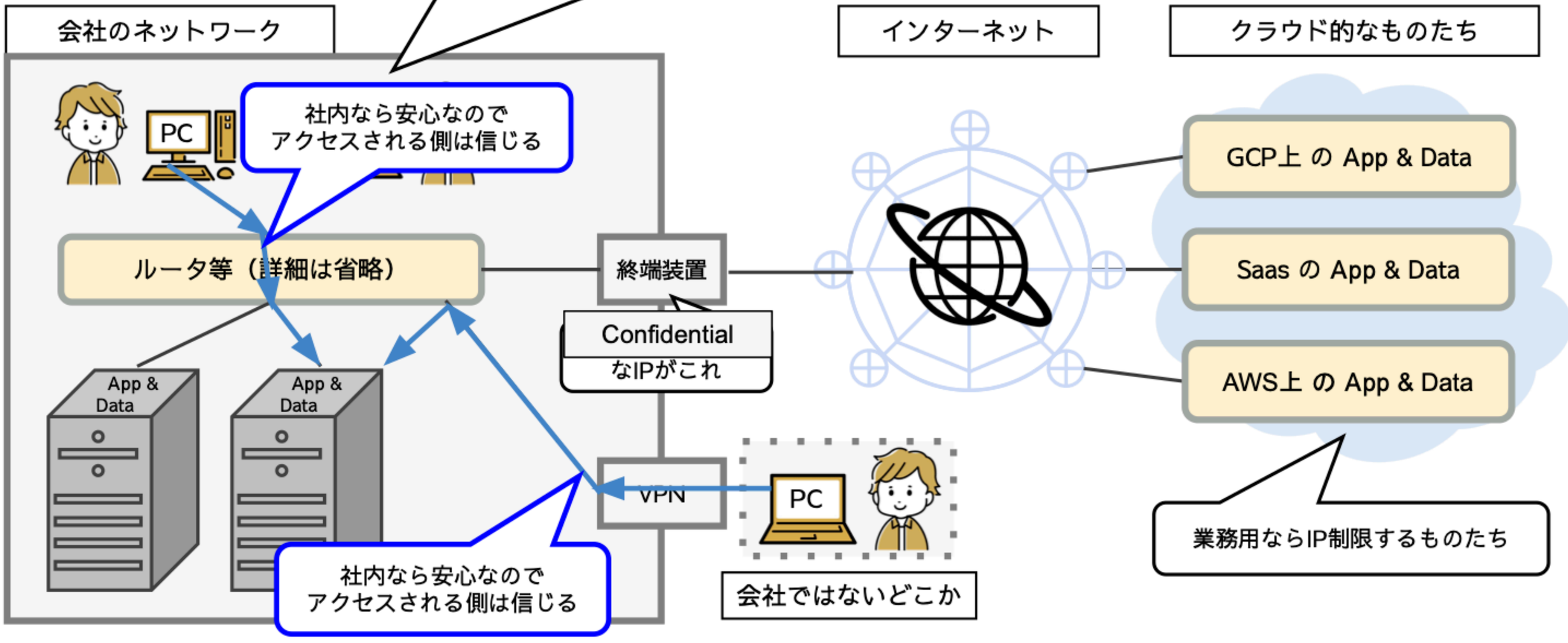
ゼロトラストとは


会社のネットワーク
インターネット
クラウド的なものたち
PC
PC
ルータ等（詳細は省略）
終端装置
App &
Data
App &
Data
VPN
PC
会社ではないどこか
GCP上 の App & Data
Saas の App & Data
AWS上 の App & Data

ここに繋ぐと認証等の各種対策/制約により下記が得られていた。

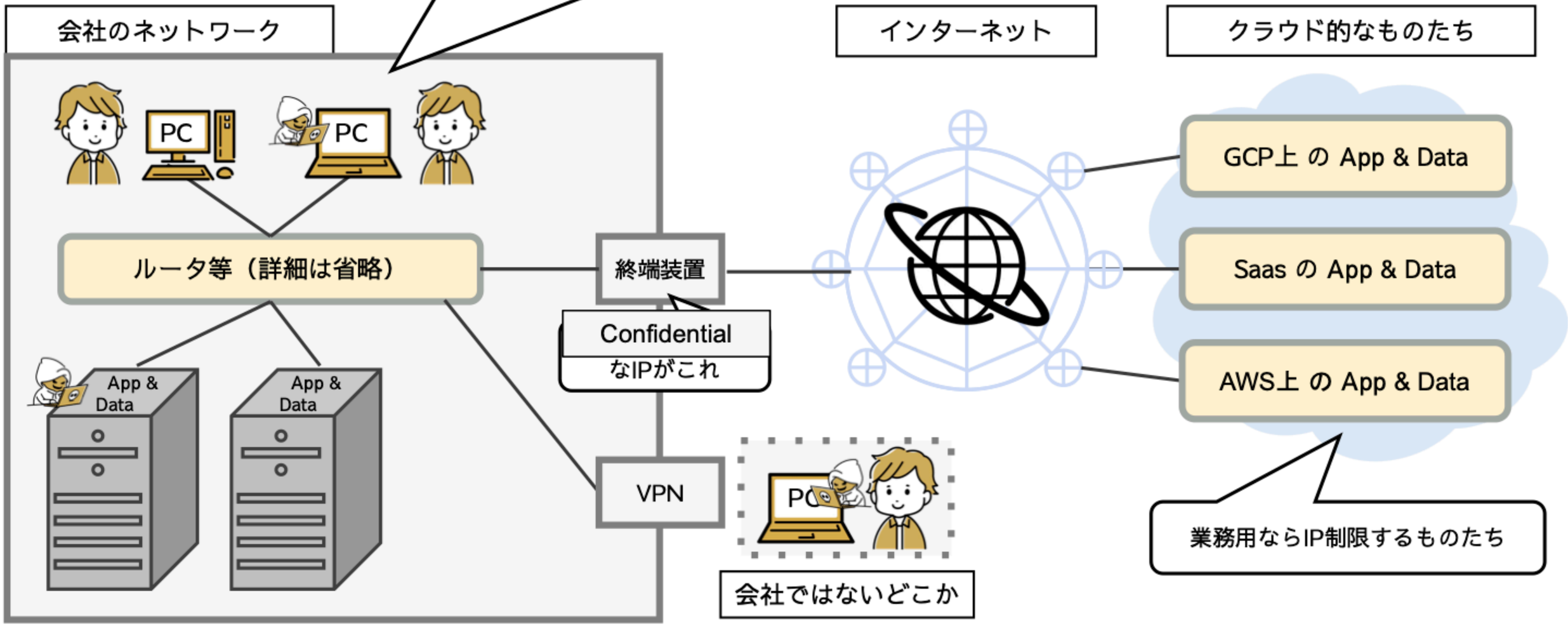
- 見知らぬ者が社内NWに居ない。→（盗聴者や攻撃者の排除）
- 攻撃監視やロギングが一定される。→（攻撃の検知や調査）
- 会社の終端装置から外に出る。→（IP制限による攻撃者排除）

社内ネットワークなら安心という世界観



ここに繋ぐと認証等の各種対策/制約により下記が得られていた。

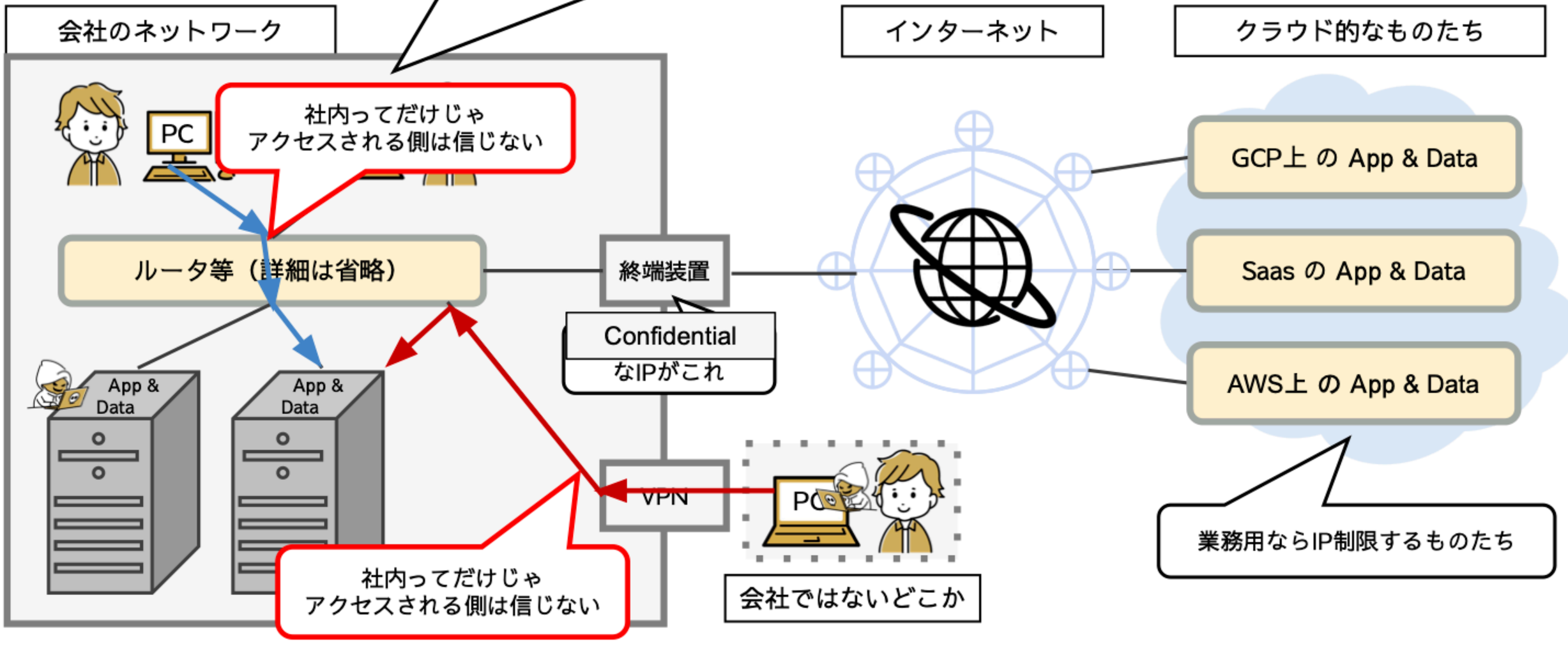
と思ったらインターネット社会では思い通りにはならなかった。
標的型攻撃はじめ色々な攻撃リスクが顕在化してきた。
色々繋がるネット社会において完全クリーンにすることは困難。



ここに繋ぐと認証等の各種対策/制約により下記が得られていた。

と思ったらインターネット社会では思い通りにはならなかった。
標的型攻撃はじめ色々な攻撃リスクが顕在化してきた。
色々繋がるネット社会において完全クリーンにすることは困難。

**社内ネットワークだからといって
それだけで安心とは考えない世界観**



ゼロトラストにおける安心とは何なのか学ぶには

MIXI
CONFIDENTIAL

2020年8月に公開された「NIST SP 800-207」が、オフィシャルな概念を勉強するのに参考になる。

▽英語

- https://csrc.nist.gov/publications/detail/sp/800-207/final

▽日本語

- https://www.ipa.go.jp/security/publications/nist/index.html

オフィシャルなゼロトラスト

MIXI
CONFIDENTIAL

遵守すべきこと

1. すべてのデータソースとコンピューティングサービスは**リソースと見なす**
2. **ネットワークの場所に関係なく**、全ての通信を保護する
3. 企業リソースへのアクセスは、**セッション単位で付与する**
4. リソースへのアクセスは、クライアントID、アプリケーション、要求する資産の状態、その他の行動属性や環境属性を含めた**動的ポリシーによって決定する**
5. 企業は、全ての資産の整合性とセキュリティ動作を**監視し、測定する**
6. 全てのリソースの認証と認可は動的に行われ、**アクセスが許可される前に厳格に実施する**
7. 企業は、資産やネットワークインフラストラクチャ、通信の現状について可能な限り多くの情報を収集し、それを**セキュリティ対策の改善に利用する**

引用元：https://www.pwc.com/jp/ja/knowledge/column/awareness-cyber-security/zero-trust-architecture-jp.html, (2022/5/1)

オフィシャルなゼロトラスト

MIXI
CONFIDENTIAL

遵守すべきこと

1. すべてのデータソースとコンピューティングサービスは**リソースと見なす**
2. **ネットワークの場所に関係なく**、全て
3. 企業リソースへのアクセスは、**セッ**
4. リソースへのアクセスは、クライア する資産の状態、その他の行動属性や環境属性を含めた**動的ポリシーによ**
5. 企業は、全ての資産の整合性とセキュリティ **し、測定する**
6. 全てのリソースの認証と認可は動的に行 **される前に厳格に実施する**
7. 企業は、資産やネットワークインフラストラクチャ、通信の現状について可能な限り多くの情報を収集し、それを**セキュリティ対策の改善に利用する**

引用元：https://www.pwc.com/jp/ja/knowledge/column/awareness-cyber-security/zero-trust-architecture-jp.html, (2022/5/1)

オフィシャルなゼロトラスト

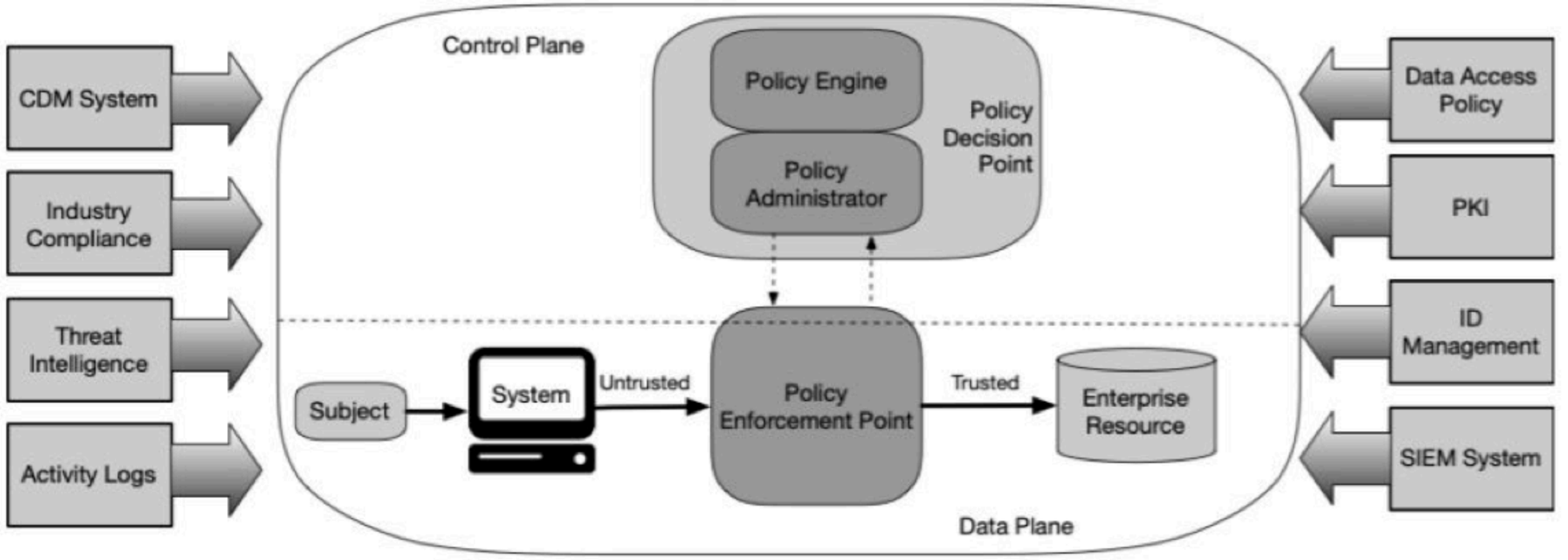


Figure 2: Core Zero Trust Logical Components

引用元：https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-207.pdf, (2022/5/1)

オフィシャルなゼロトラスト

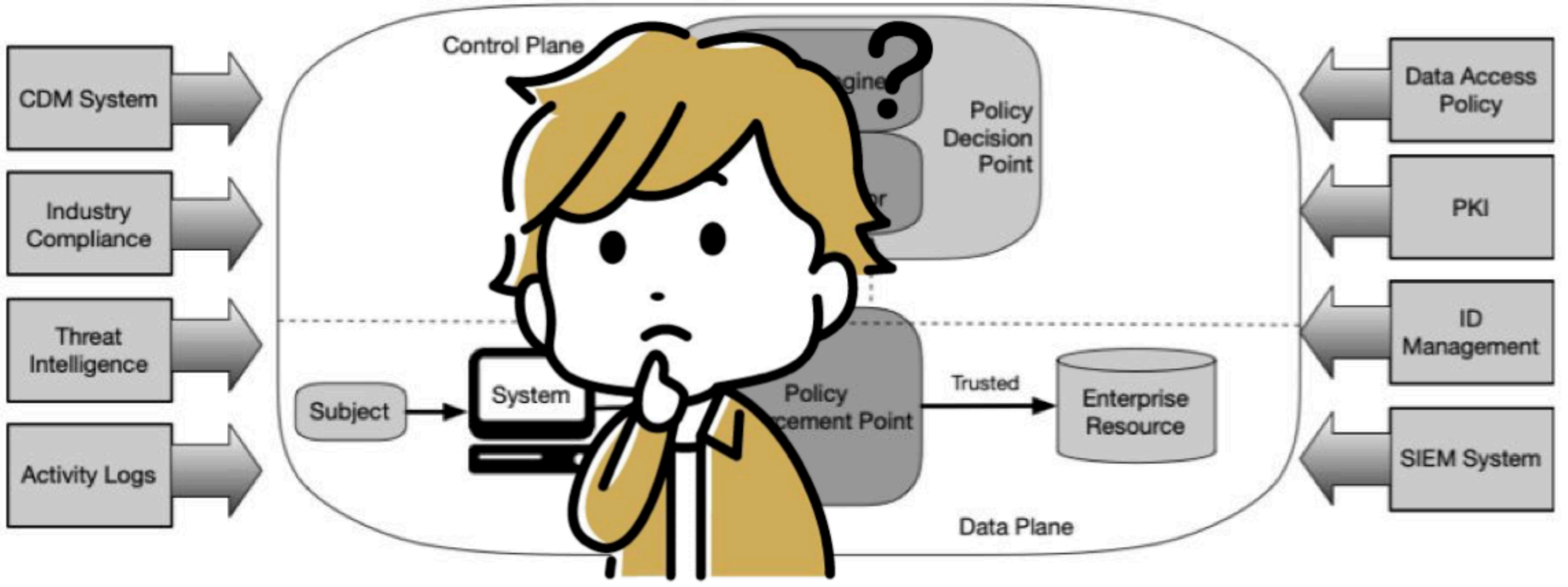


Figure 2: Core Zero Trust Logical Components

引用元：https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-207.pdf, (2022/5/1)

立ち返っておきたいこと

ゼロトラストという概念を正確に理解すること自体が大事なのではない。

- 境界防御の何が問題で、何をしたくて
  -
- そのためにゼロトラストを用いればどう解決ができるのか
  -

立ち返っておきたいこと

ゼロトラストという概念を正確に理解すること自体が大事なのではない。

- 境界防御の何が問題で、何をしたくて
  - 「社内NWなら安全」という神話の崩壊が問題で、もっときめ細やかなアクセス制御をしたい。
- そのためにゼロトラストを用いればどう解決ができるのか
  - ゼロトラストには色々な概念が登場するが、「エンフォーサ」という概念を実現することで上記を解決することができる。

エンフォーサとは？

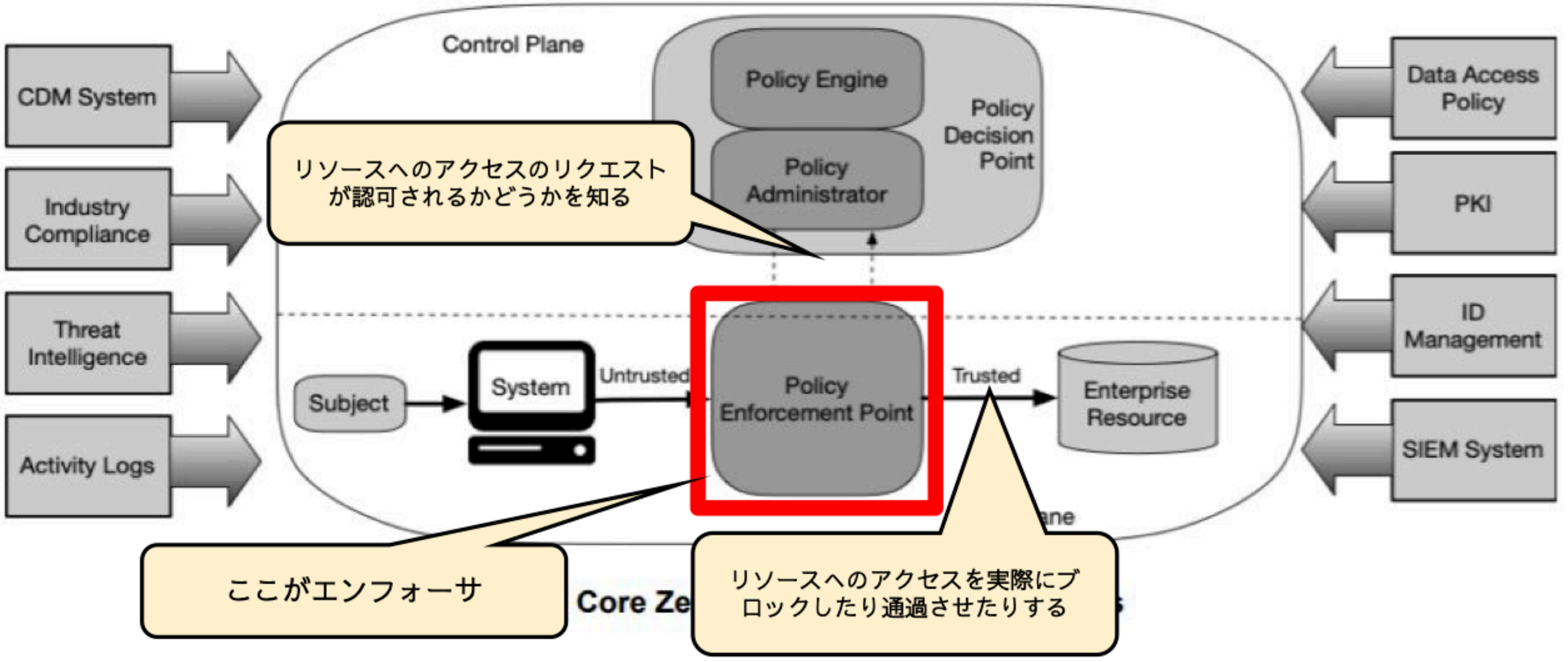
MIXI
CONFIDENTIAL

Untrustedな領域とtrustedな領域を分かつ、境界線となるコンポーネントのこと。

- エンフォーサの2つの役割
  - Policy Decision Point（信頼できるリクエストか判断するポイント）とのやり取り
    - リソースへのアクセスのリクエストが認可されるかどうかを知る。
  - 認可の判断の実際の導入と継続的な適用
    - リソースへのアクセスを実際にブロックしたり通過させたりする。

エンフォーサとは？

- コンポーネントの図で言うところは下記にあたる。



引用元：https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-207.pdf, (2022/5/1)

エンフォーサとは？

- エンフォーサが「通過させていいか」知るための材料。

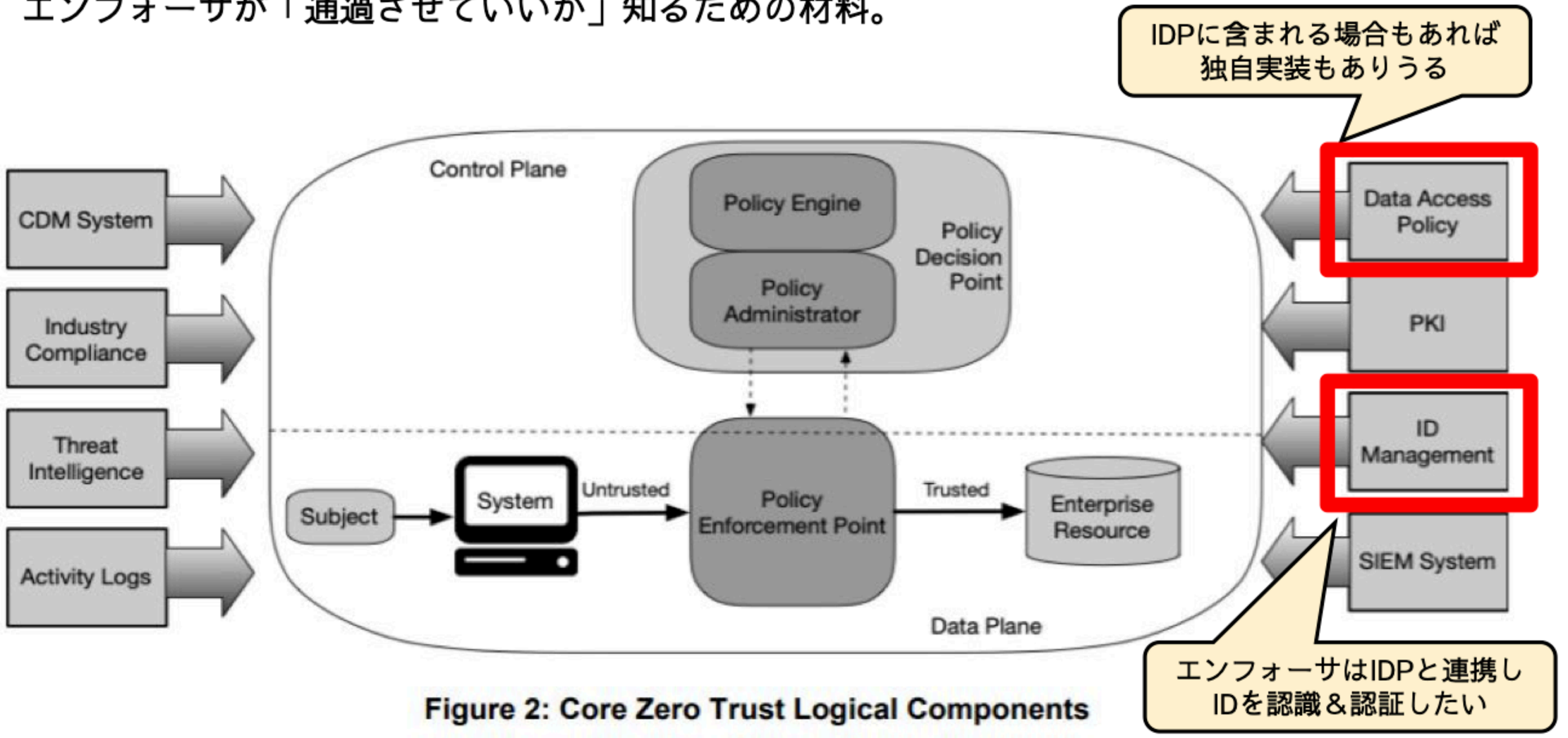
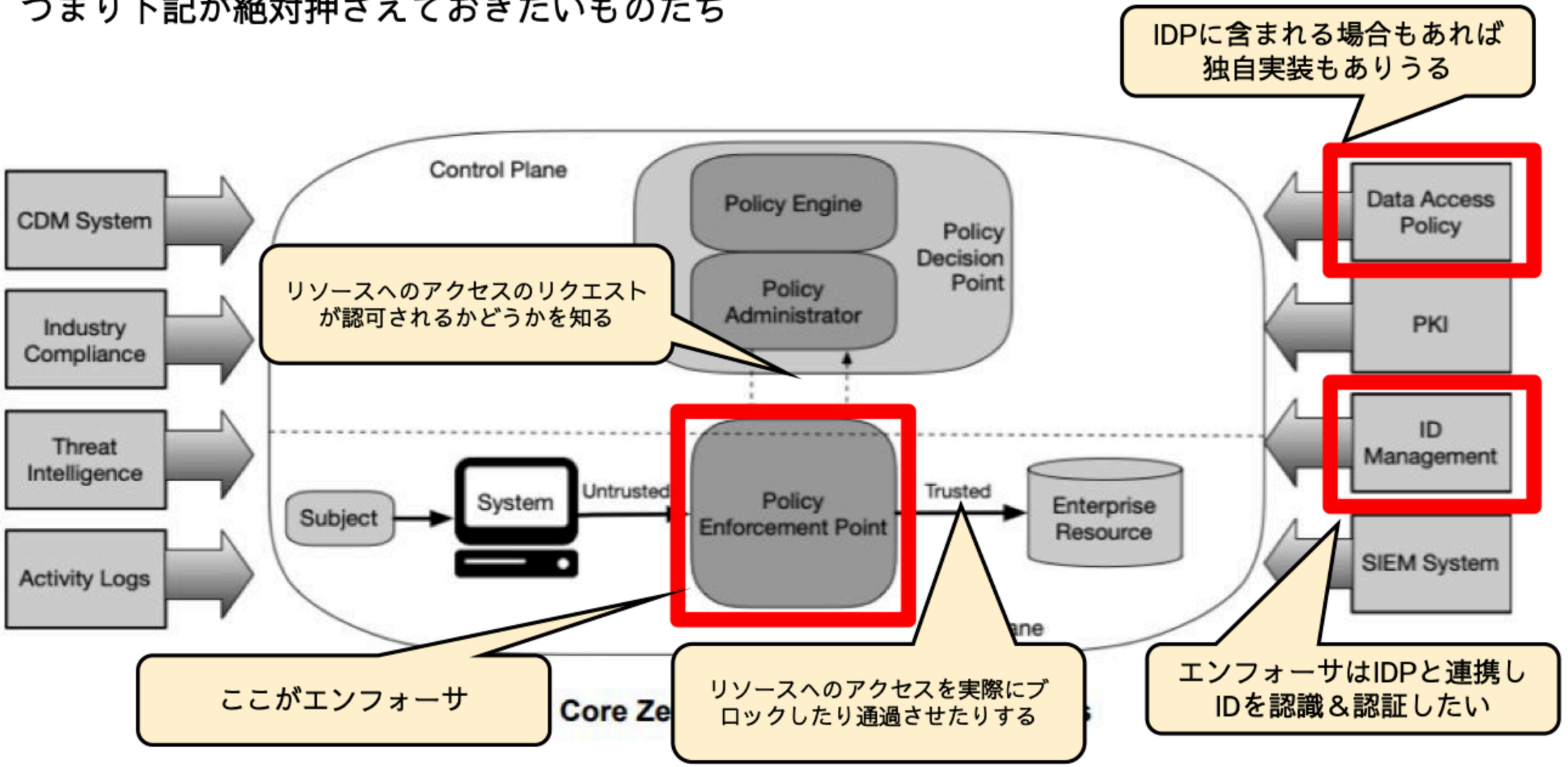


Figure 2: Core Zero Trust Logical Components

引用元：https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-207.pdf, (2022/5/1)

エンフォーサとは？

- つまり下記が絶対押さえておきたいものたち



引用元：https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-207.pdf, (2022/5/1)

エンフォーサがなぜ重要か

MIXI
CONFIDENTIAL

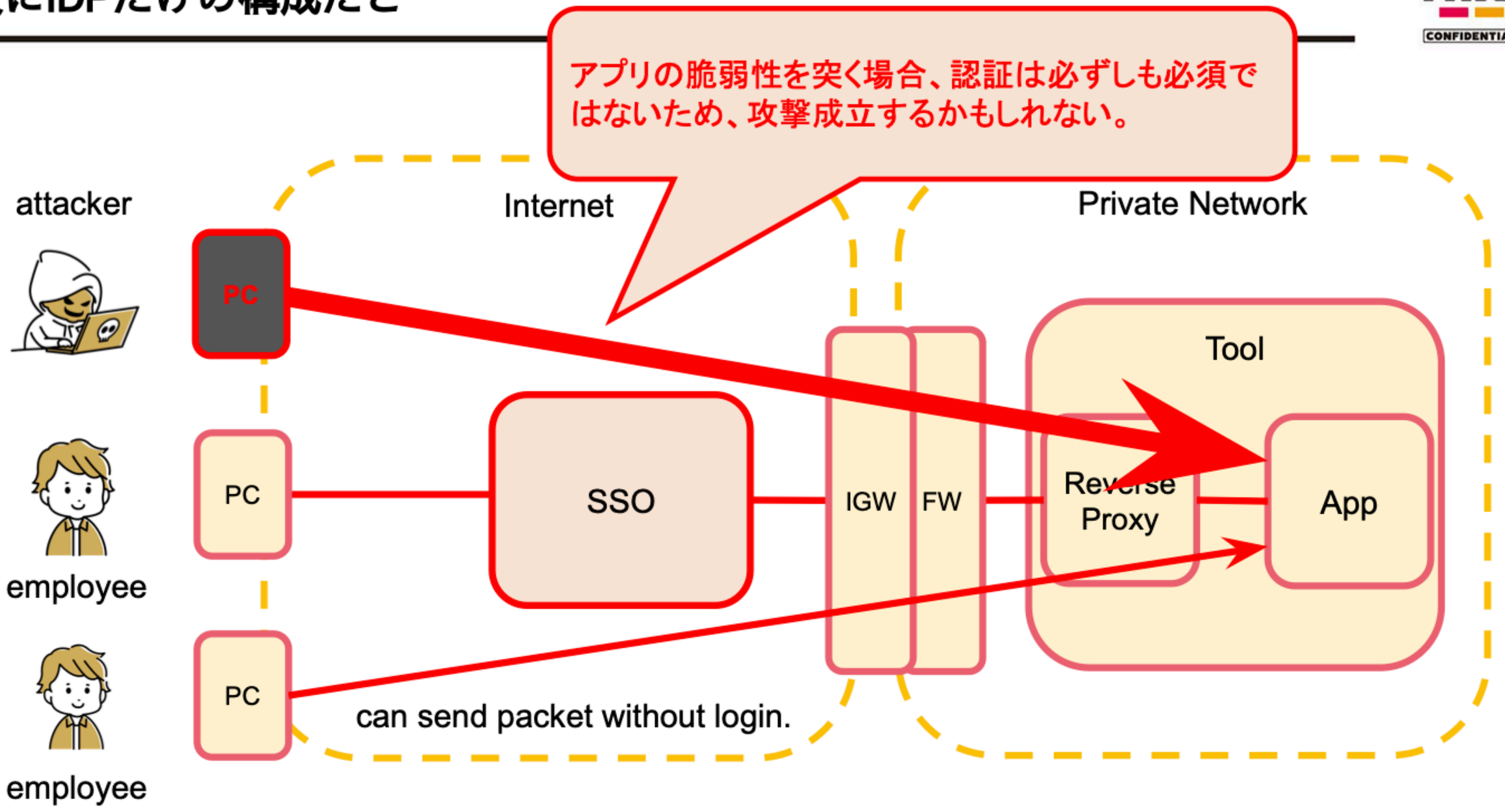
ゼロトラスト実装にあたってmustなこと

- 少なくとも今までの境界防御より弱くなることがあってはいけない。

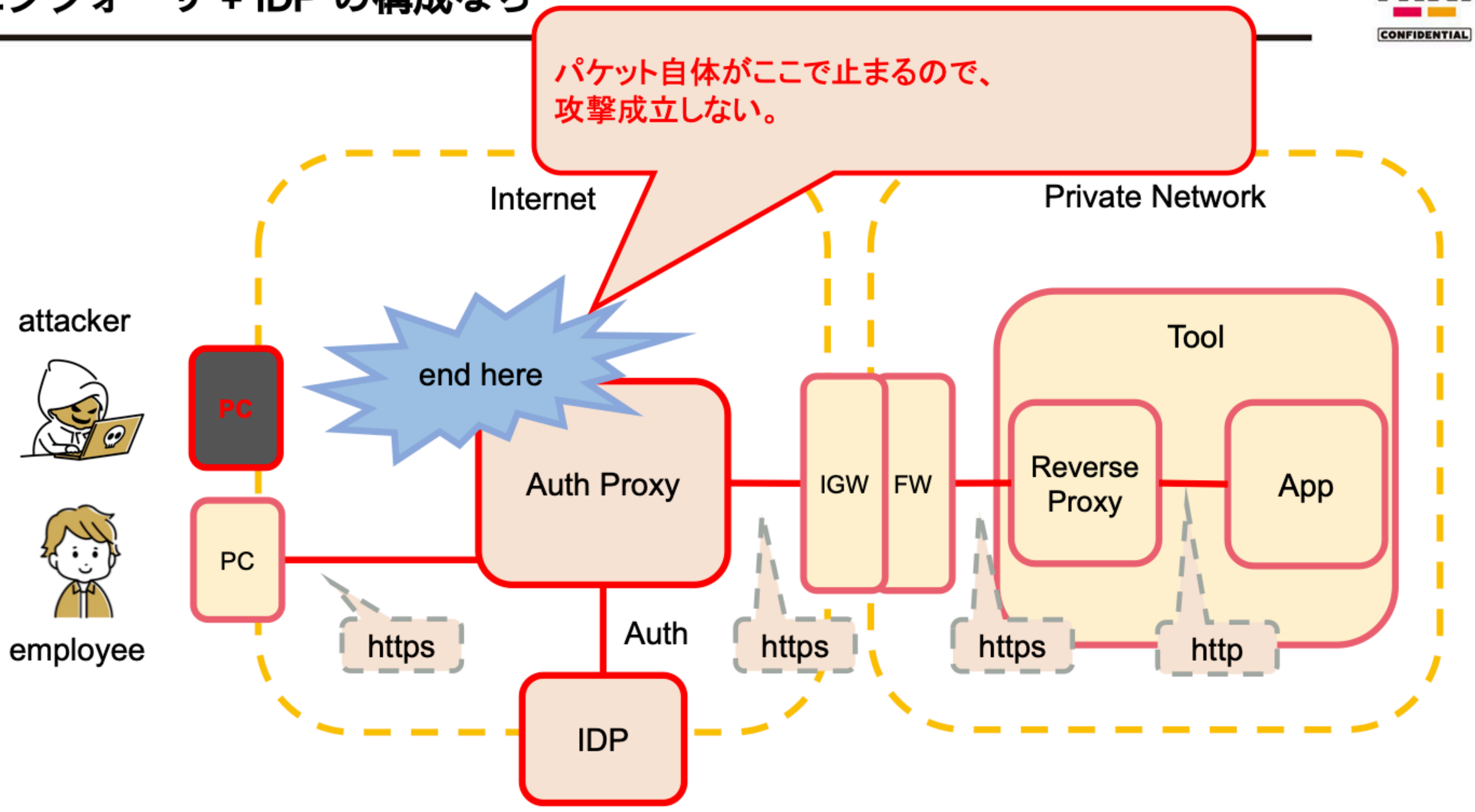
- 境界防御より弱くならずに済むためには、
  - アクセス制御対象への**全て**のリクエストをチェックする必要がある。
    - 具体的にはプロキシやアプリケーションロードバランサを利用したい。
- つまりIDPやアプリ本体の認証**だけ**だと不十分。
  - 攻撃リクエスト自体はリソースを持つアプリに直接届いてしまうため。
  - アプリの脆弱性を突いた攻撃には十分な防御効果を発揮できないため。
  - チェックできないリクエストが存在するため。
- だからユーザ認証が行えて全通信で認証と許可のチェックができるようにする必要がある。
  - 出来れば信頼のおけるIDPと連携すると良い。

エンフォーサによってこれらを満たすことができる。
だからエンフォーサは大事。

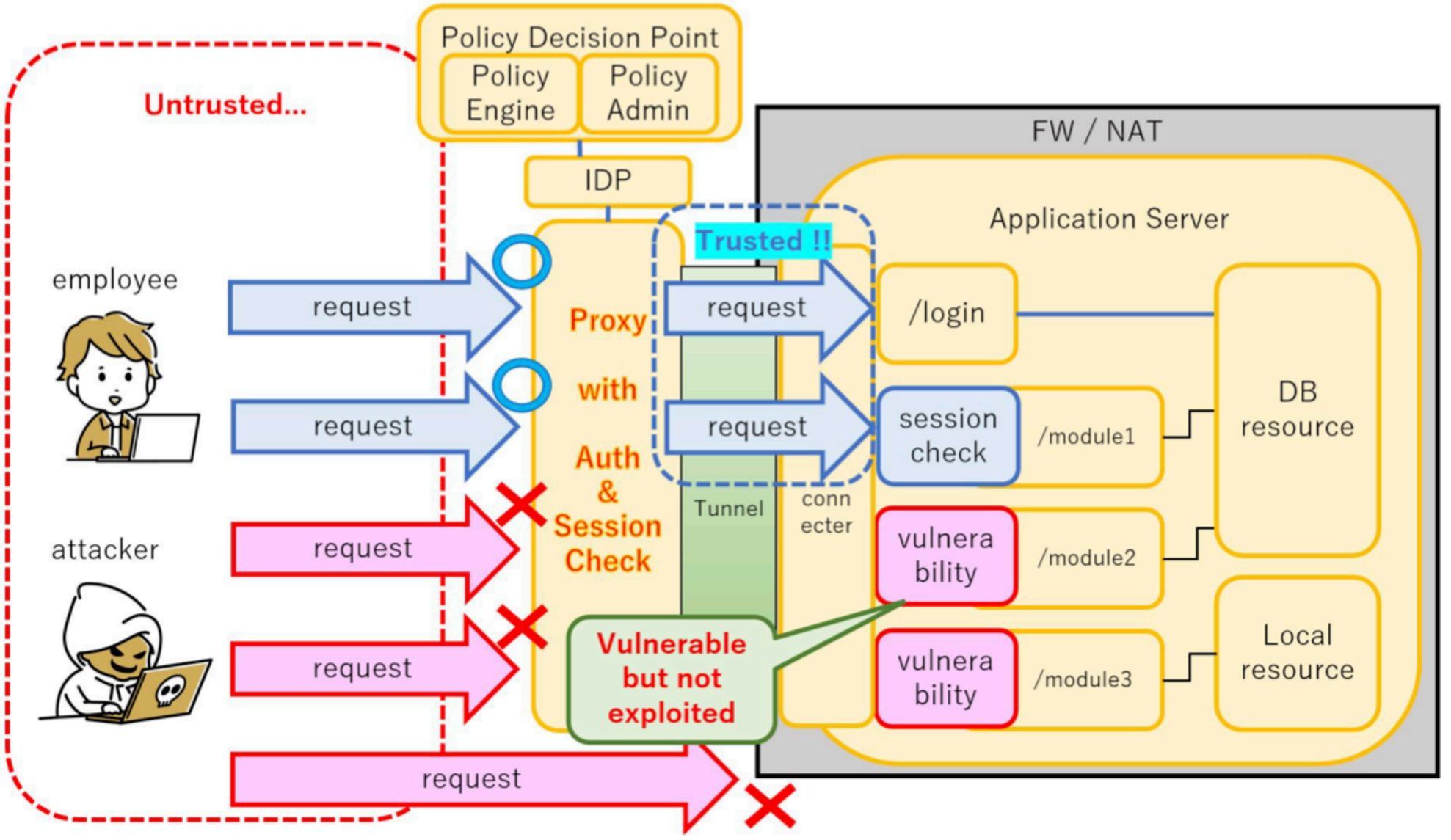
仮にIDPだけの構成だと



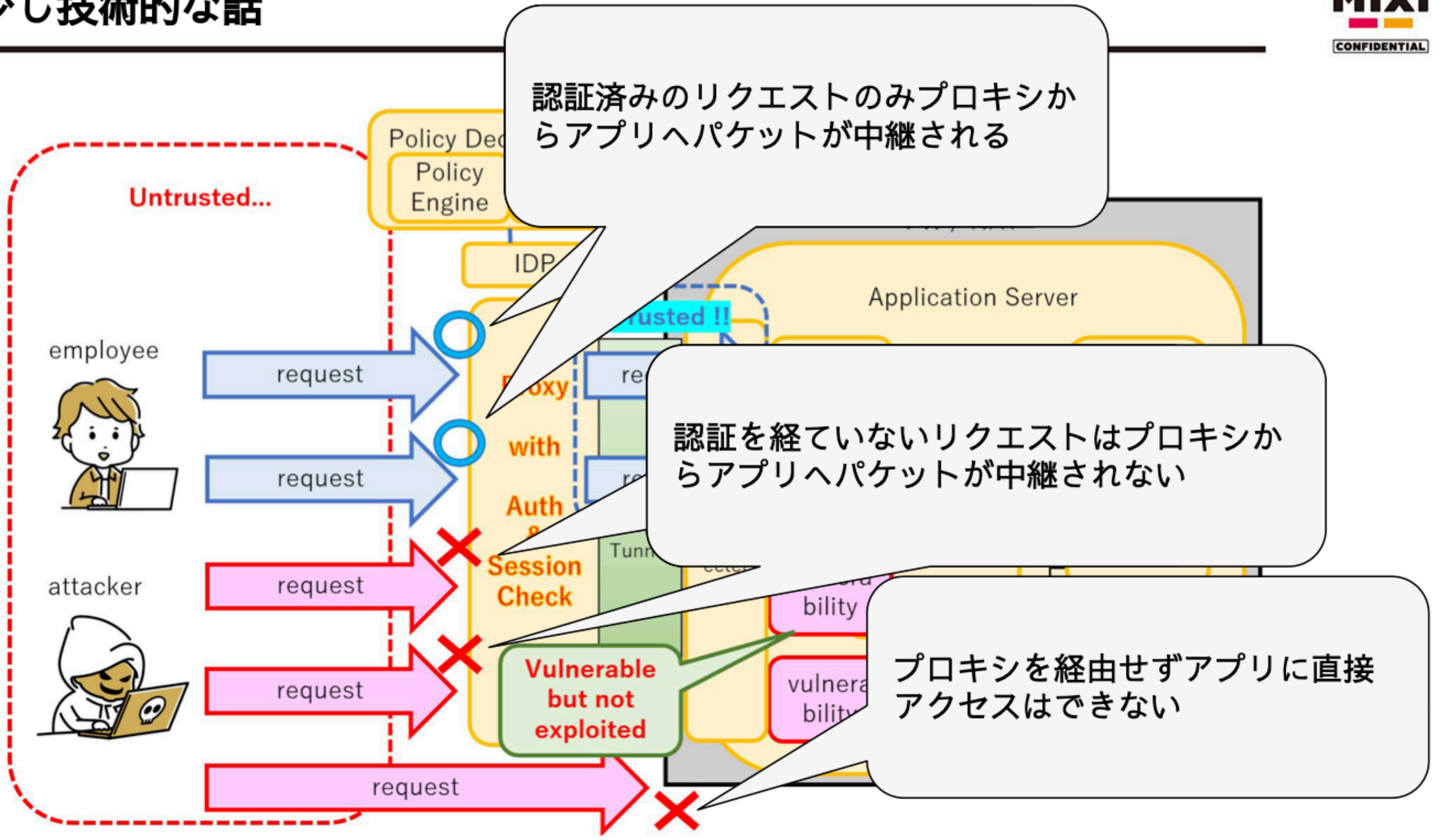
エンフォーサ + IDP の構成なら



もう少し技術的な話


Policy Decision Point
Policy Engine
Policy Admin
Untrusted...
IDP
FW / NAT
Application Server
Trusted !!
employee
attacker
request
request
request
request
request
Proxy with Auth & Session Check
Tunnel
conn ecter
/login
session check
/module1
vulnera bility
/module2
vulnera bility
/module3
DB resource
Local resource
Vulnerable but not exploited

もう少し技術的な話


認証済みのリクエストのみプロキシからアプリへパケットが中継される
Untrusted...
Policy Dec
Policy Engine
IDP
Application Server
employee
request
request
Proxy with Auth & Session Check
認証を経ていないリクエストはプロキシからアプリへパケットが中継されない
attacker
request
request
Vulnerable but not exploited
プロキシを経由せずアプリに直接アクセスはできない
request

結局のところゼロトラストは

MIXI
CONFIDENTIAL

「社内ネットワークだからというだけで信用しない」
「色々と疑いの目をもってアクセス制御する」
これに尽きる。

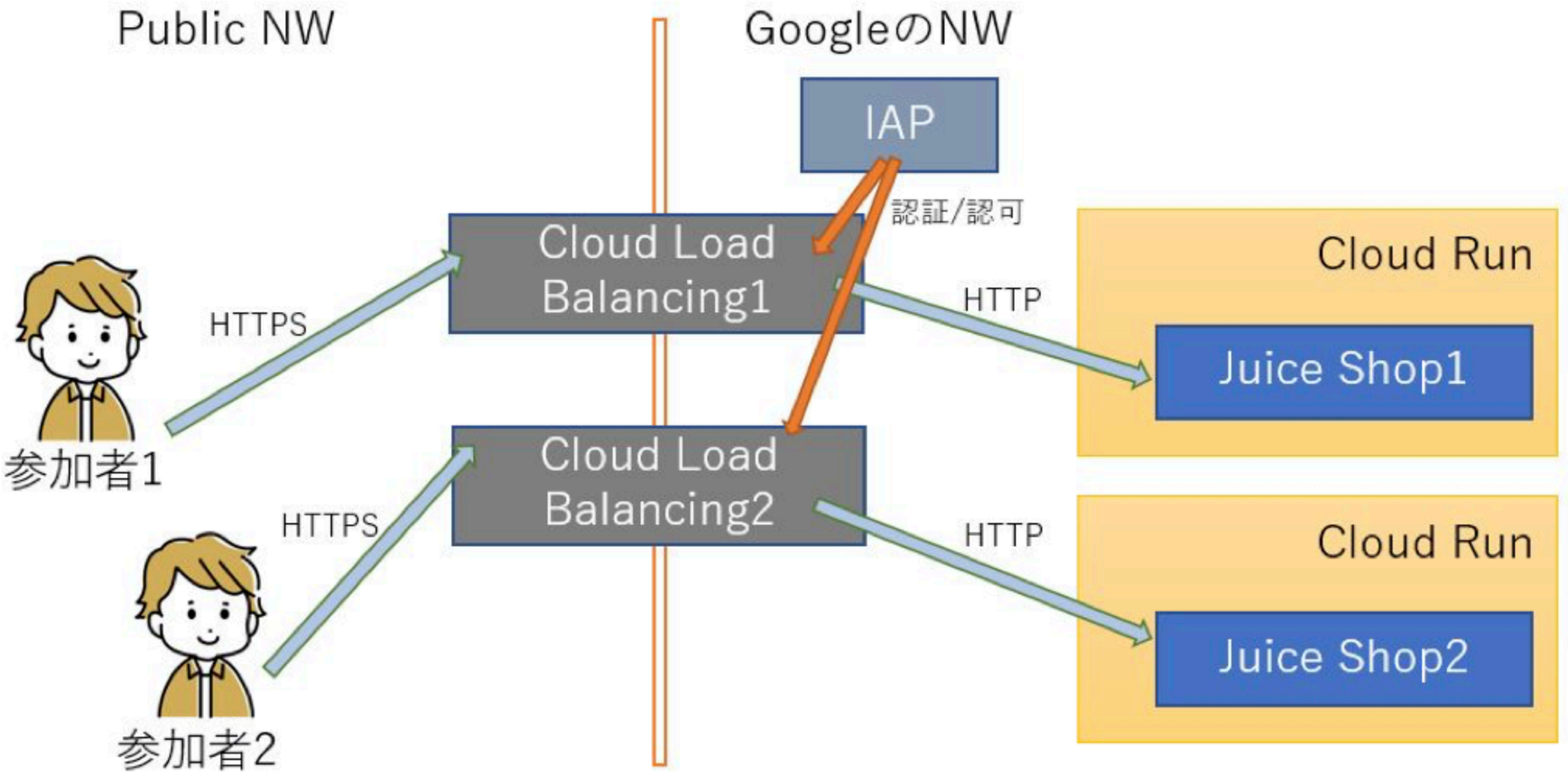
そのためのモデルや計画や概念。

いきなり教科書通り100点満点の実装じゃなくてもOK。
ただし実装において絶対に抑えておくべきポイントはある。
それが先述のエンフォーサ。

何から何を守るのか。何のためにやるのか。が大事。

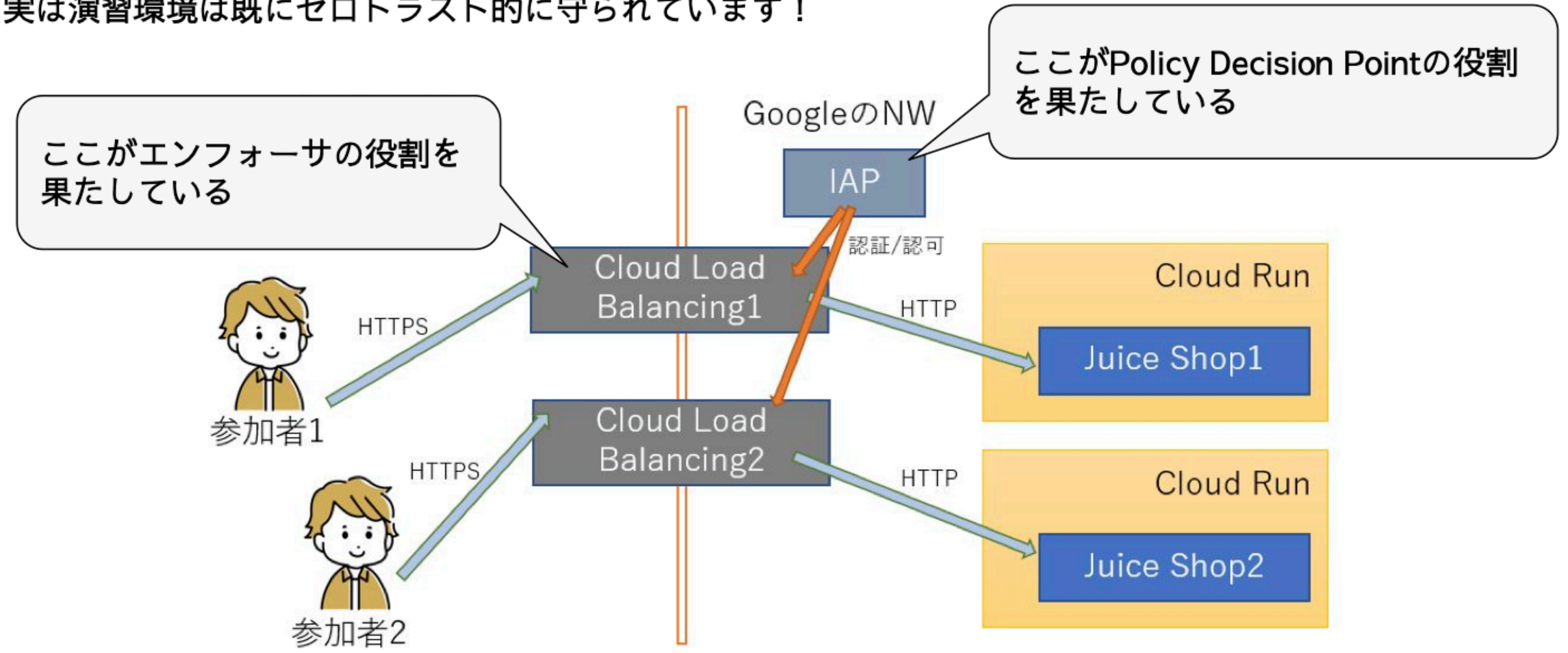
Juice Shopをゼロトラスト的に守るとしたら？

皆さんの演習環境はこのようになっています。



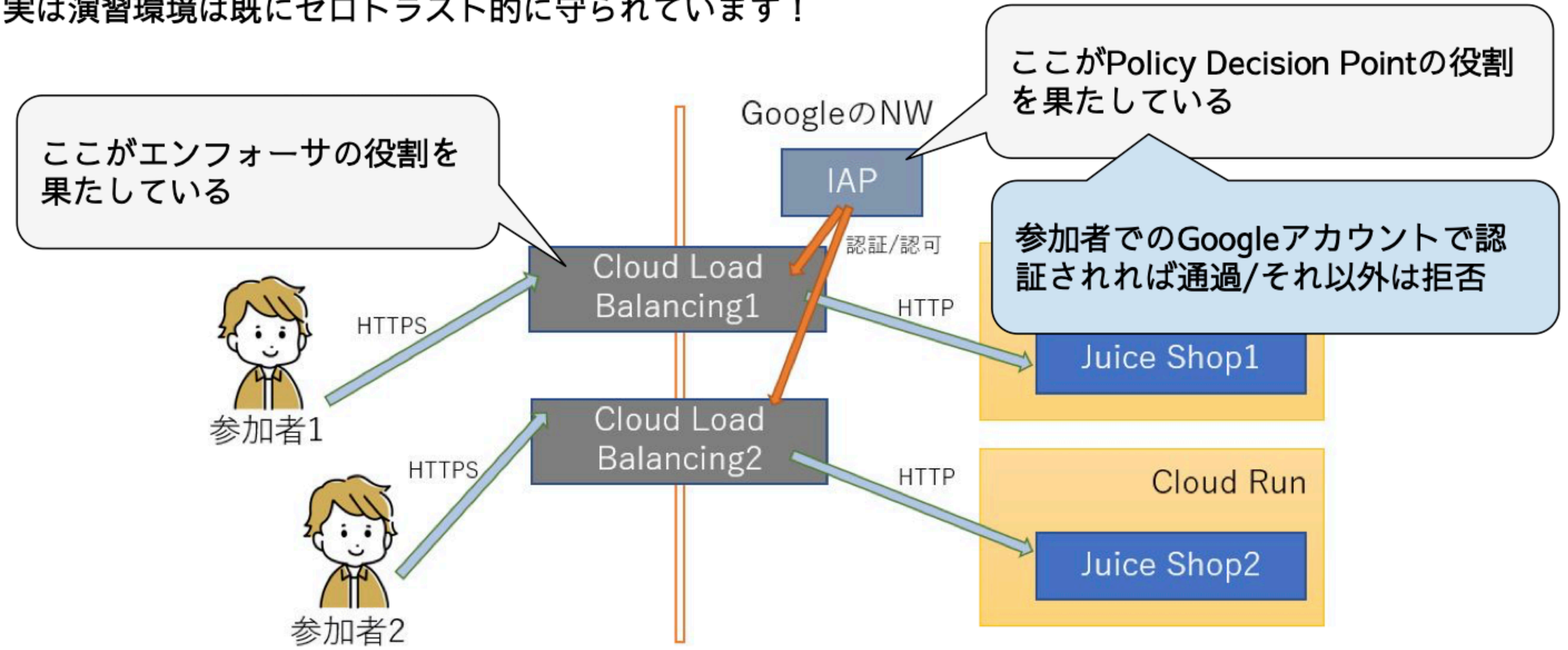
Juice Shopをゼロトラスト的に守るとしたら？

実は演習環境は既にゼロトラスト的に守られています！



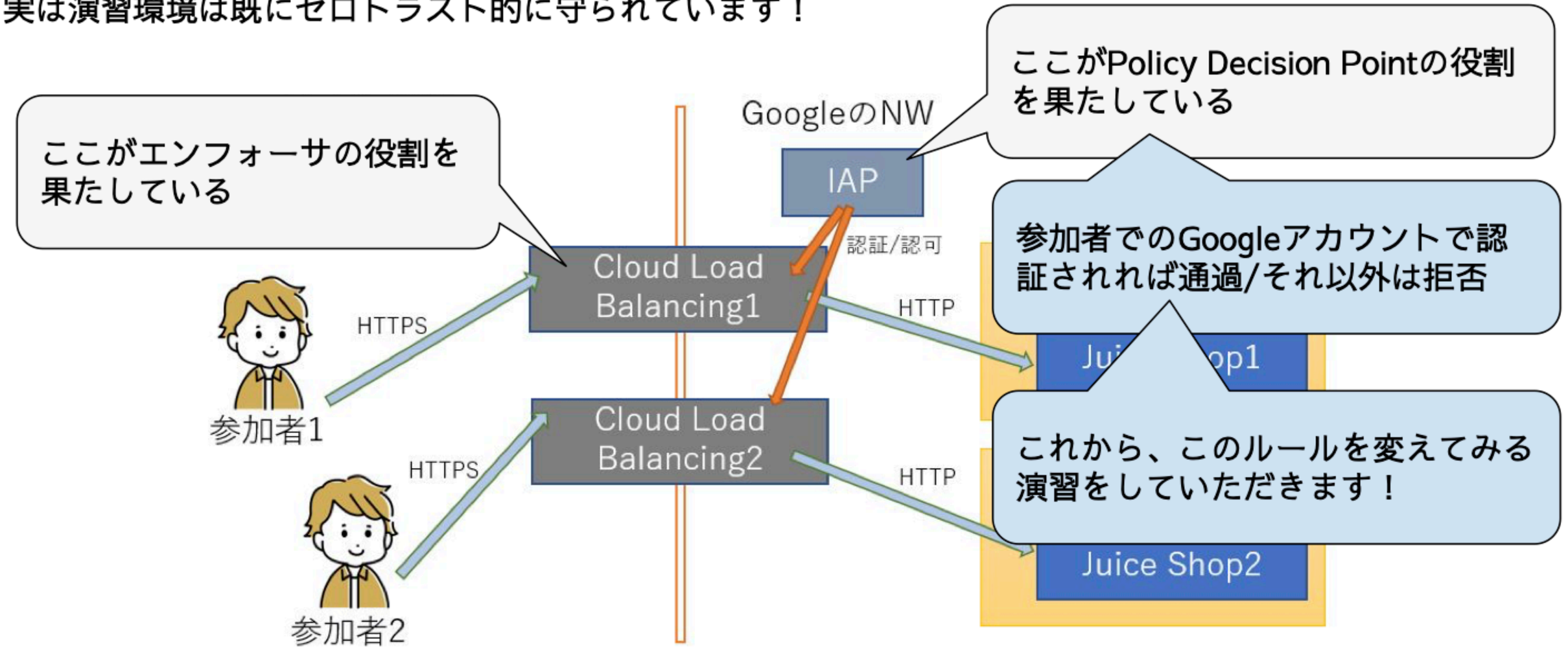
Juice Shopをゼロトラスト的に守るとしたら？

実は演習環境は既にゼロトラスト的に守られています！



Juice Shopをゼロトラスト的に守るとしたら？

実は演習環境は既にゼロトラスト的に守られています！



【演習】Juice Shopをゼロトラストにもとづいて守ろう

自身のJuice Shop環境のIAP設定を操作し、ゼロトラストにもとづいたアクセス制御を行おう。

- IAPとは
  - IAP とは、ウェブサイトへのリクエストをインターセプトし、リクエストを送信したユーザーを認証して、認証されたユーザーにのみサイトへのアクセスを許可する、という一連の処理を行うサービスです。
  - 対応しているプラットフォームは、App Engine、Compute Engine のほか、Google Cloud ロードバランサの背後で動作するサービスなど、さまざまなものがありますが、その利用は Google Cloud に限定されません。IAP コネクタと合わせて使用すれば、オンプレミスのアプリケーションを保護することも可能です。
  - ざっくりいうと、ロードバランサ等へのアクセス時にGoogleアカウントの認証を挟めるサービス。

【演習】Juice Shopをゼロトラストにもとづいて守ろう

MIXI
CONFIDENTIAL

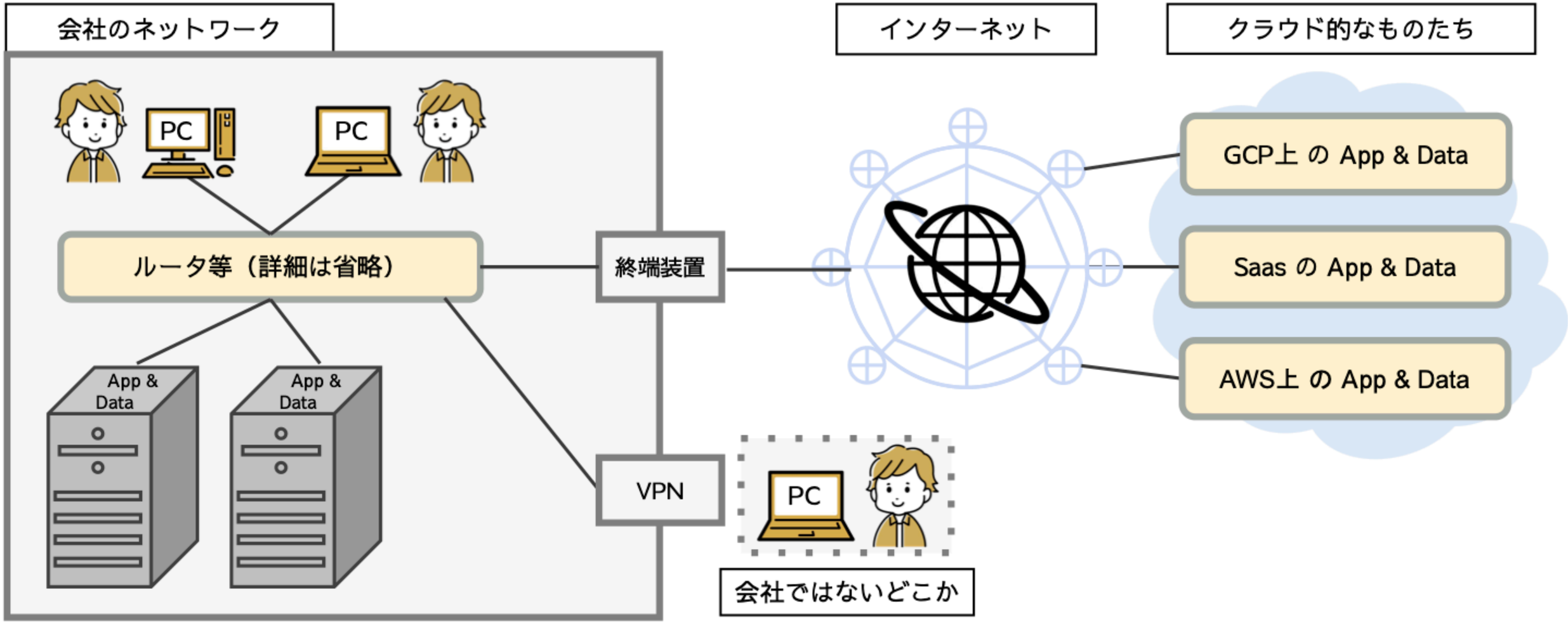
自身のJuice Shop環境のIAP設定を操作し、ゼロトラストのアクセス制御を行おう。

やること

- 下記手順書に則って進めてみましょう。
  - Confidential

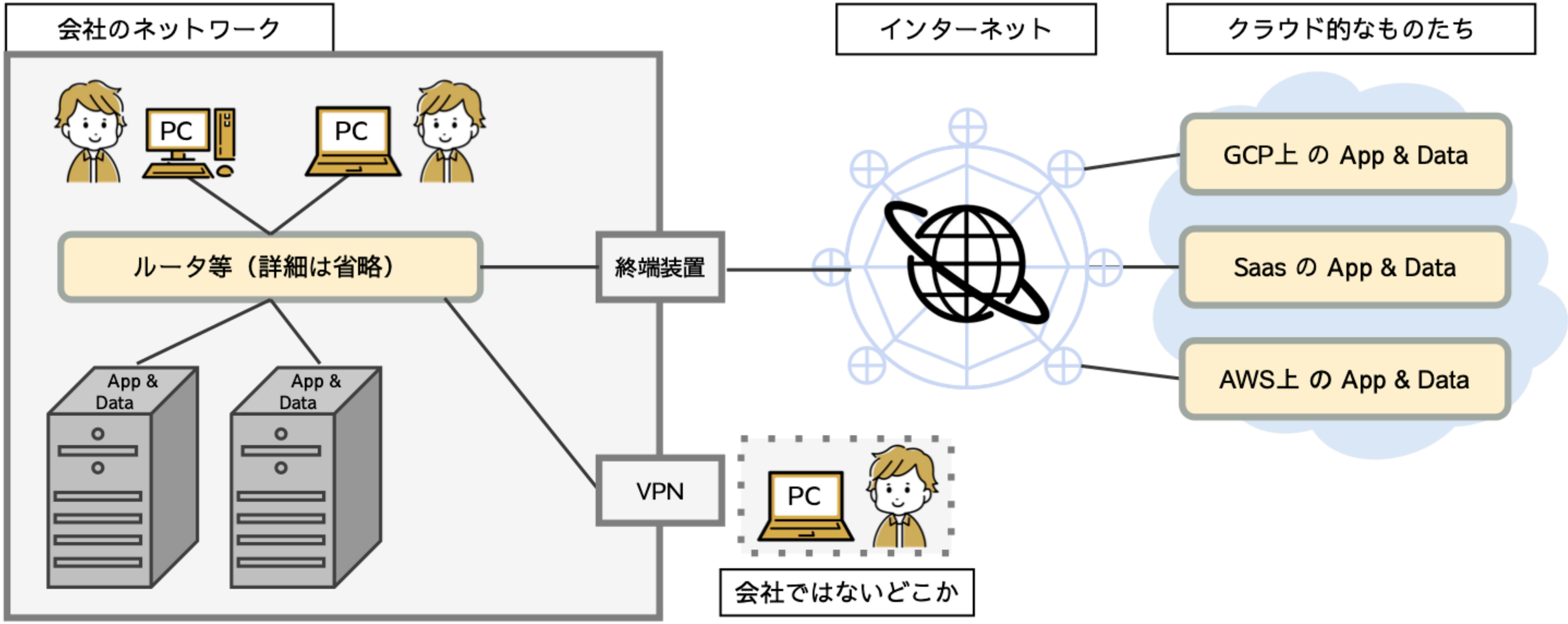
- 確認すること
  - IAPをONにして、かつアクセス権があるときの挙動
    - 普通にアクセスできればOK
  - IAPをONにして、かつアクセス権が無いときの挙動
    - レスポンスが「You dont have access」になればOK
  - IAPをOFFにしたときの挙動
    - シークレットブラウザなど、Googleログインしてない状態でアクセスできたらOK
    - 先ほど施したIP制限はアリのままで行ってください！
      - IAPオフでIP制限も無いと、演習環境がインターネット公開されることになってしまうため。

ミクシィ社とゼロトラスト



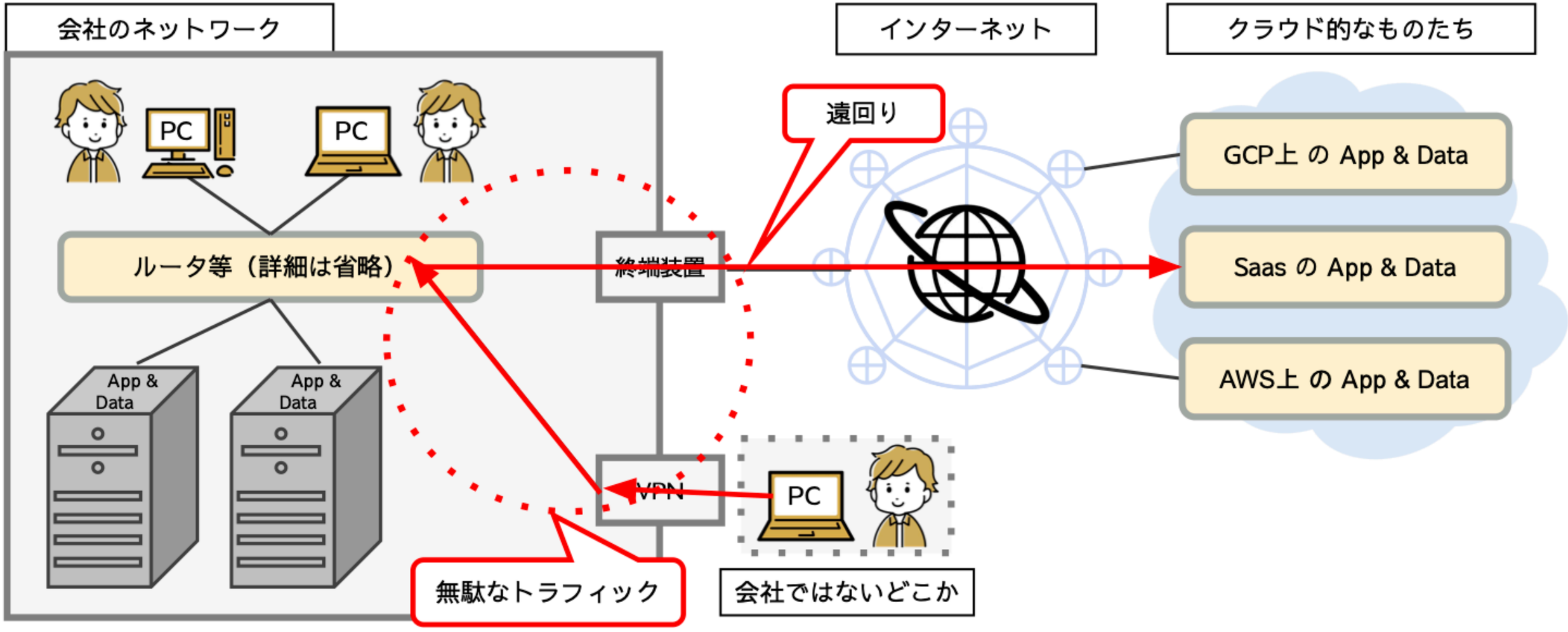
コロナによるリモートワークの普及には、いくつか問題があった。

- VPNクライアントの都度起動/切替が面倒臭すぎる問題。
- VPNだとネットワーク的に遠回り問題。
- VPNを全員常時使う想定じゃなかった問題。キャパシティ。
- 社内ネットワークよりも攻撃者の発見や除外が困難。



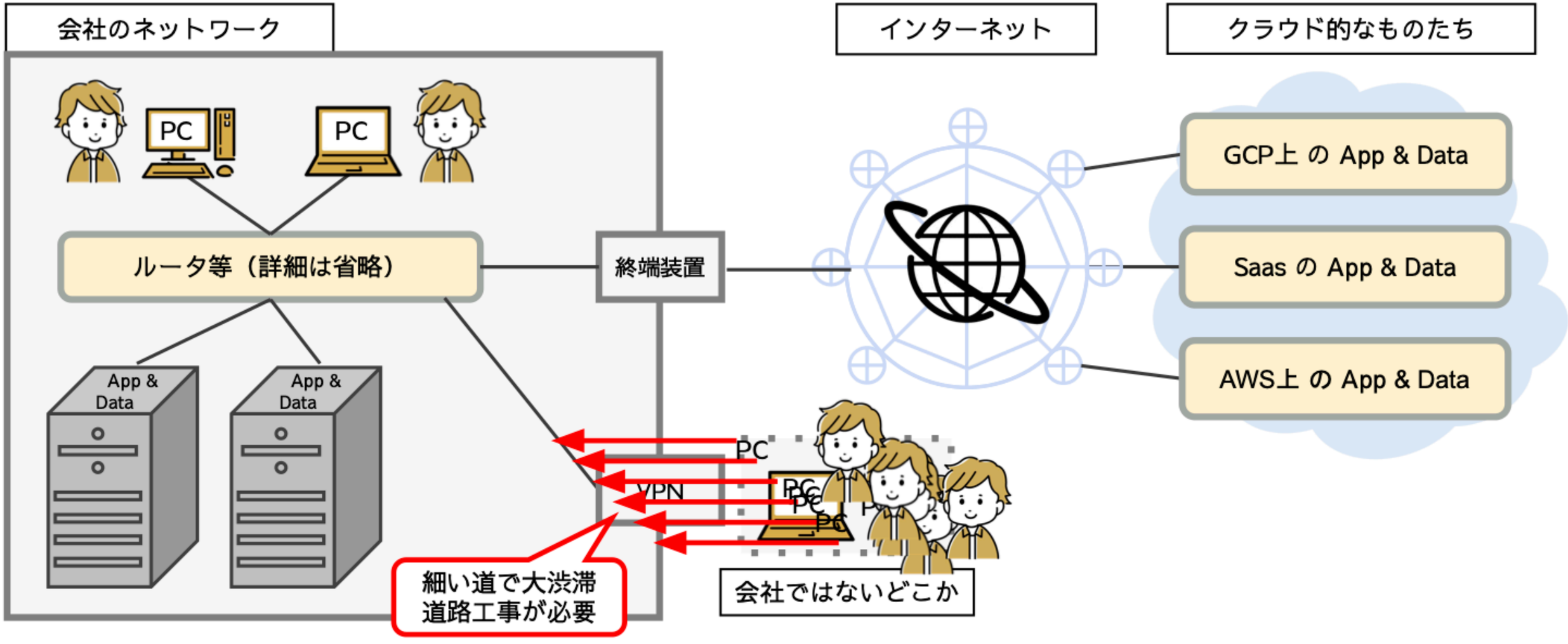
コロナによるリモートワークの普及には、いくつか問題があった。

- VPNクライアントの都度起動/切替が面倒臭すぎる問題。
- **VPNだとネットワーク的に遠回り問題。**
- VPNを全員常時使う想定じゃなかった問題。キャパシティ。
- 社内ネットワークよりも攻撃者の発見や除外が困難。



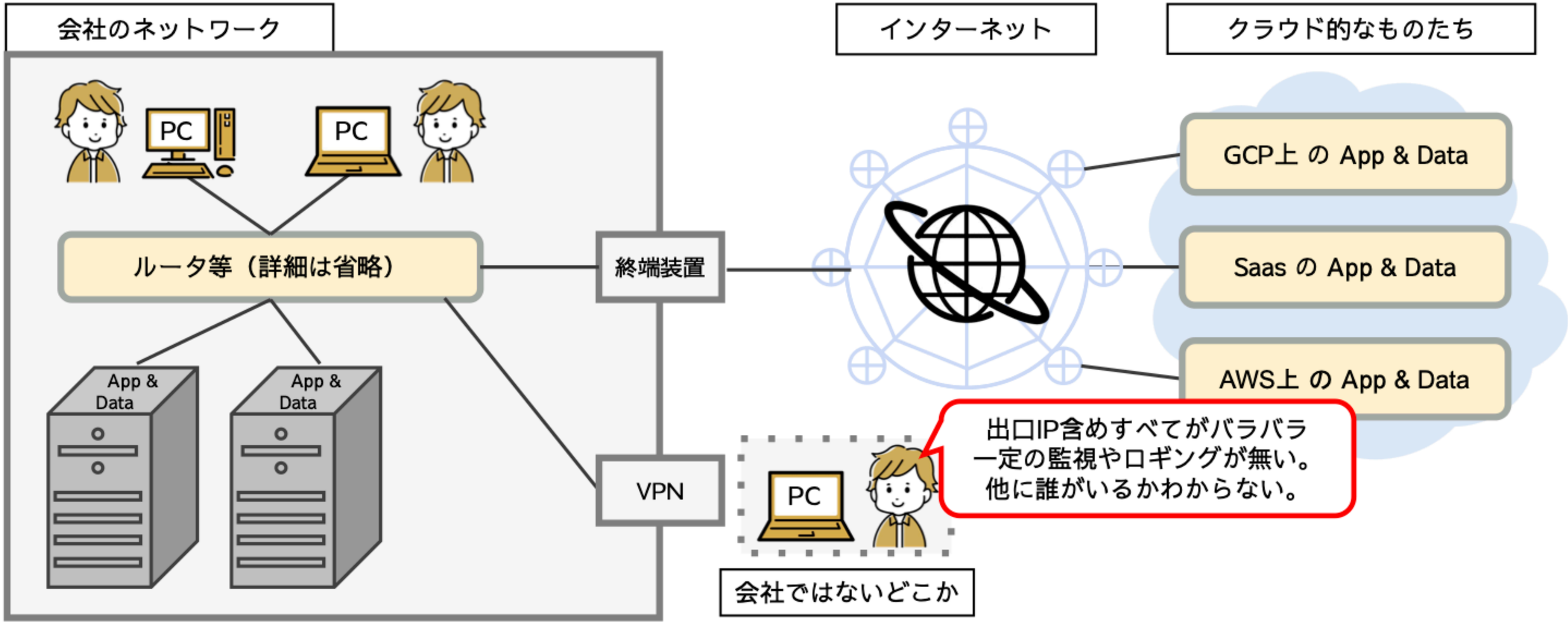
コロナによるリモートワークの普及には、いくつか問題があった。

- VPNクライアントの都度起動/切替が面倒臭すぎる問題。
- VPNだとネットワーク的に遠回り問題。
- **VPNを全員常時使う想定じゃなかった問題。キャパシティ。**
- 社内ネットワークよりも攻撃者の発見や除外が困難。



コロナによるリモートワークの普及には、いくつか問題があった。

- VPNクライアントの都度起動/切替が面倒臭すぎる問題。
- VPNだとネットワーク的に遠回り問題。
- VPNを全員常時使う想定じゃなかった問題。キャパシティ。
- **社内ネットワークよりも攻撃者の発見や除外が困難。**



ミクシィ社とゼロトラスト

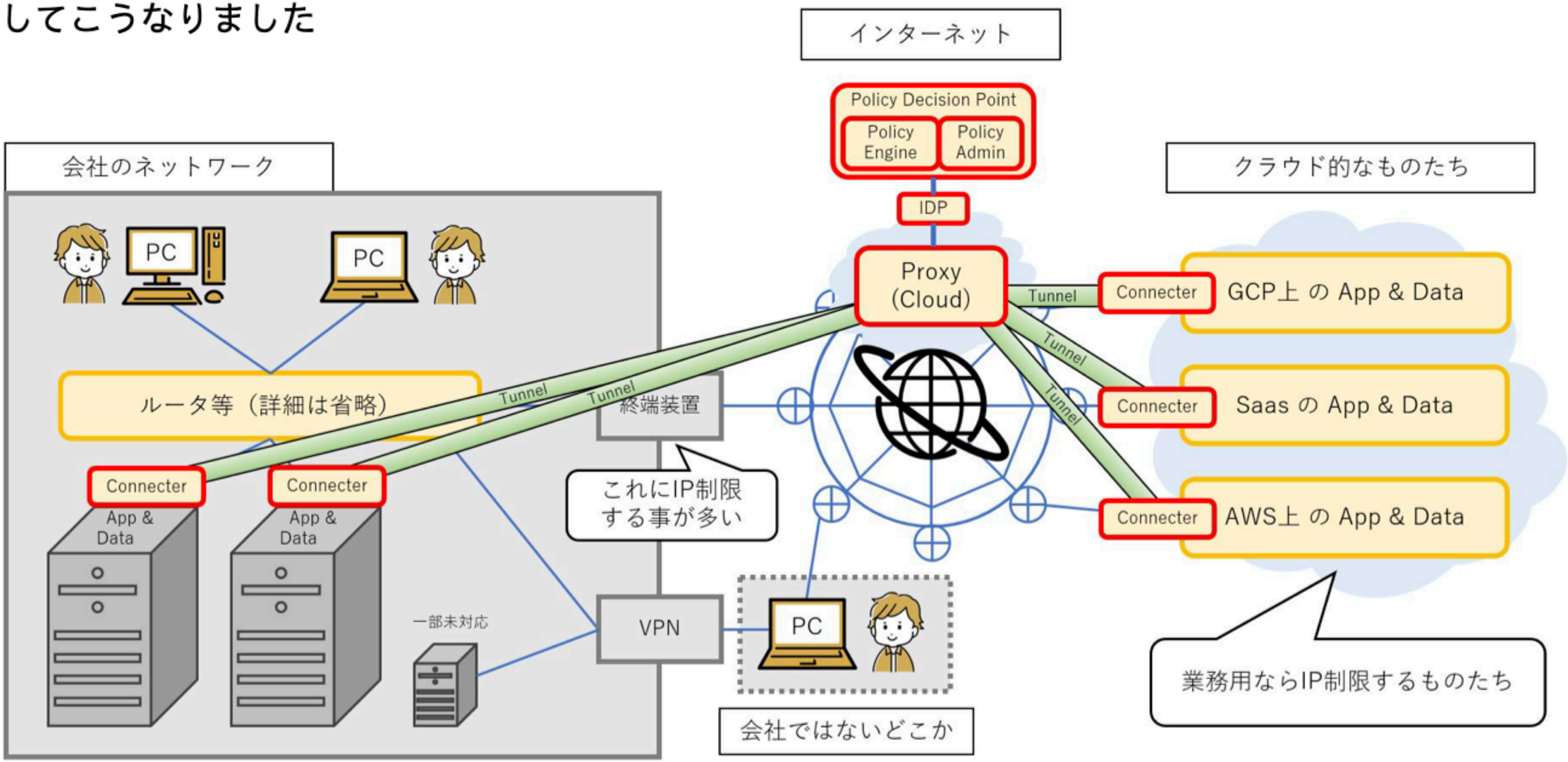
昨今のセキュリティと働き方の状況は、ゼロトラストはどちらにとっても都合が良かった。

- 境界防御だけじゃ防ぎきれないという時流の変化。（セキュリティ）
- コロナでリモートワークせざるをえないという時流の変化。（働き方）

ミクシィ社とゼロトラスト

MIXI
CONFIDENTIAL

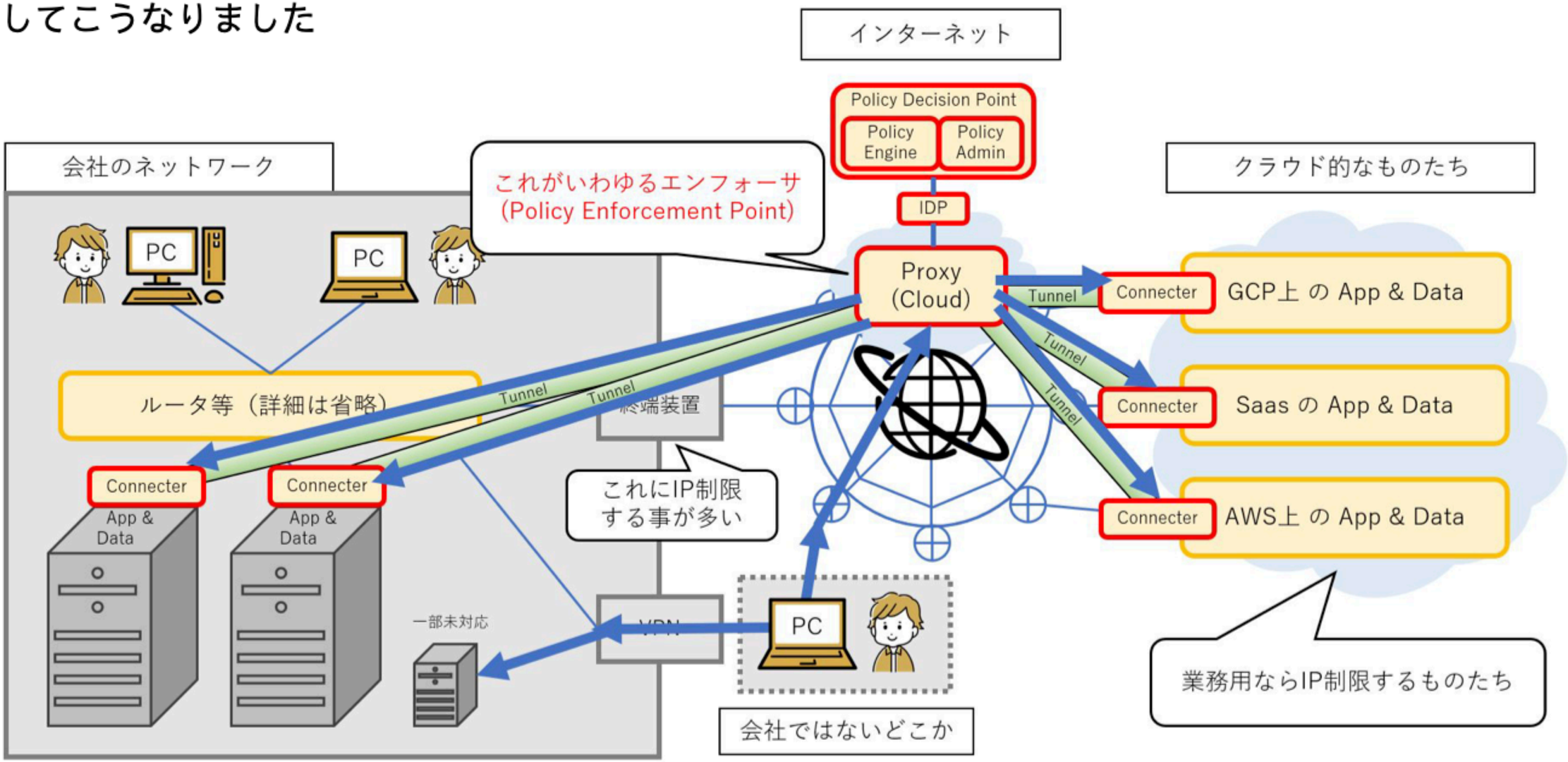
そしてこうなりました



ミクシィ社とゼロトラスト

MIXI
CONFIDENTIAL

そしてこうなりました



IDPに Policy Decision Point が寄っているワケ

MIXI
CONFIDENTIAL

プロキシはあくまで通信を通過/遮断させているだけで、通過可否の判定役はIDP側に寄っている。
それはゲートウェイであるプロキシと疎結合にすることで、下記のようなメリットを得るため。

- SSOの利便性と全体的なセキュリティ底上げの相乗効果。
  - アカウントの棚卸しなど管理工数削減。
  - CSIRTやSOCとの連動のしやすさによるセキュリティレベル担保。
- ゼロトラストの実装のしやすさとセキュリティ強化。
  - 部署や環境によって使いたいゲートウェイが異なる。
    - 例えば　AWS→ALB　GCP→IAP　オンプレ→EAA　のように。
    - ゲートウェイは各環境それぞれ、極力リソースの近くに配備しておくのがセキュア。
      - というかこれは必須レベル。遠いと境界防御感が増すため。

（餅は餅屋。認証は認証屋。）

ゼロトラストの取り組み状況

MIXI
CONFIDENTIAL

- 非公開システム（やIP制限したくなるようなSaaS）が主な対象。
- 主要な社内システムについては既にVPN無しでリモートアクセス可能。
- 事業ごとの非公開システムについては啓発や支援を通じて推進。
  - ドキュメントや勉強会によるナレッジ共有。実装作業。自分達でやっている事業もある。
  - 脆弱性診断やIaaS監視をきっかけに誘導する事がある。
- 全てのケースでゼロトラストが必須というよりは制限の選択肢の1つ。

今後IDPで実現できると一層きめ細やかになるポイント

MIXI
CONFIDENTIAL

- 2FA（出来れば物理か生体）
- リスクベース認証（普段と違う時に再認証）
- デバイス認証（クライアント証明書）
- ゼロトラスト的なデバイスチェック（エージェントの提供）

これらができると、
より強固なセキュリティを均一に提供できる。

ユーザが誤解しがちなこと

MIXI
CONFIDENTIAL

プロキシ等でトンネルする部分が外からは見えづらいせいかもしれないが、
下記のような誤解がたまにありそう。

- IDPの認証が付いていればVPN要らないんだ。へー。
  - 例えば社内ツールにアクセスする際IDPログインが求められるので、
    IDPに目が行きがちになるかもしれないが、大事なのは全パケットを前段で弾くこと。
  - 弾くためにIDPを利用しているだけ。併せてエンフォーサが必要。

- カフェやホテルのフリーwifiで仕事できるじゃん。
  - リモートから社内ツールにセキュアにアクセスできるようになったのでそう思ってしまうかもしれないが、
    カフェ等での業務やフリーWi-Fiのリスクは、境界防御/ゼロトラスト的な観点以外にもある。
    - 盗聴やのぞき見など
  - 結局のところ目的や業務にあわせた対策や状況確認が当分は必要。
  - ゼロトラスト・E2Eの暗号化・PC防御が100%対応済みなら別かも。

設計者が誤解してはならないこと

MIXI
CONFIDENTIAL

- 境界防御の全てがオワコンみたいな事ではない。
  - 範囲や使い方は違えど依然として使われているし、重要。
  - 具体的にはFW/ネットワーク接続時の認証など。

- 単純に ゼロトラスト = リモート何でもOK には現状ならない。
  - 改めてPC側の保護も必要となる。AV、EDR、PFW、覗き見、etc...
  - バラバラなネットワークセキュリティに耐えられる設計が必要。

ゼロトラストを誤解/過信して、
セキュリティレベルが下がることは無いようにしなければいけない。

【演習】(余談) Juice ShopをWAFで守ろう

MIXI
CONFIDENTIAL

先ほど皆さんに触っていただいたCloudArmorには、WAF機能があります。
それを適用してみましょう。

【演習】(余談) Juice ShopをWAFで守ろう

MIXI
CONFIDENTIAL

WAFについて

- Webアプリケーションの脆弱性に対しては一つ一つ対策を施していくのが基本だが、WAF(WebApplicationFirewall)によって防御するという選択肢もある。
  - リクエスト中に攻撃パターンの文字列が含まれていると遮断する(403応答など)。
  - WAFだけで守り切るのは困難だったり、正常なリクエストにも反応してしまう可能性があったり、**基本的にはおすすめはしていないが、特定のケースで使える場合があるため、選択肢として持っておくとよい。**
  - レガシーなつくりでコードが入り組んでおり、アプリケーションの改修が難しいといった場合、WAFで一元的に攻撃シグネチャを遮断するという対策手段はアリかもしれない。
- GCPではCloud ArmorにWAF機能がある。

- CloudArmorのWAF機能の参考Docbase記事
  - Confidential

総合演習

MIXI
CONFIDENTIAL

3. SQLインジェクションを悪用して、全ユーザの情報（Eメール、パスワードハッシュ）を抜き出そう

答え（再掲）

- 商品検索機能に下記のようなリクエストを送信することで、SQLインジェクションにより商品テーブルとUsersテーブルの情報が結合され、ユーザ情報を取得できます。

- /rest/product/search?q=apple%27))%20UNION%20SELECT%20email,password,3,4,5,6,7,8,9%20FROM%20Users--
  - ※%27は「'」、%20は「 」をそれぞれURLエンコードした値

ユーザー一覧の情報を奪取できた！

総合演習

3. SQLインジェクションを悪用して、全ユーザの情報（Eメール、パスワードハッシュ）を抜き出そう

答え（再掲）

- 商品検索機能に下記のようなリクエストを送信することで、SQLインジェクションにより商品テーブルとUsersテーブルの情報が結合され、ユーザ情報を取得できます。

- /rest/product/search?q=apple%27))%20UNION%20SELECT%20email,password,3,4,5,6,7,8,9%20FROM%20Users--
  - ※%27は「'」それぞれURLエンコードした値

これをWAFで防ぐ！

【演習】(余談) Juice ShopをWAFで守ろう

MIXI
CONFIDENTIAL

自身のJuice Shop環境にWAF設定を施し、SQLインジェクションを検知・遮断しよう。

やること

- 下記手順書に則って進めてみましょう。
  - Confidential
  - 「403 Forbidden」エラーを確認できたらOK

- 確認すること
  - 通常のリクエストには何も反応しない一方、
  - 総合演習3 の攻撃クエリ
    「/rest/product/search?q=apple%27))%20UNION%20SELECT%20email,password,3,4,5,6,7,8,9%20FROM%20Users--」のような攻撃リクエストが送信されると、「403 Forbidden」エラーとなることを確認する。

サーバ側を強固にするには？

MIXI
CONFIDENTIAL

- 公開するところ
  - 自分たちの作りこむもの（皆さんがコーディングする部分）
    - → 脆弱性を知り、対策する
  - 自分たちでは作りこまないもの（ライブラリやミドルウェア等）
    - → リスクを正しく評価し、必要に応じ対策する
- **公開する必要のないところ**
  - **→ アクセス制御を行って、アクセス自体を絞る**

ここの話をした！

クライアント側について

MIXI
CONFIDENTIAL

ここからはクライアント側のセキュリティリスクについてお話していきます。

クライアント側について

MIXI
CONFIDENTIAL

API/Webと同様、モバイルにも、Owasp Mobile Top10というTop10の脆弱性がある。

- M1: Improper Platform Usage
- M2: Insecure Data Storage
- M3: Insecure Communication
- M4: Insecure Authentication
- M5: Insufficient Cryptography
- M6: Insecure Authorization
- M7: Client Code Quality
- M8: Code Tampering
- M9: Reverse Engineering
- M10: Extraneous Functionality

引用元：https://owasp.org/www-project-mobile-top-10,(2022/4/26)

ただし、これは2016年のものなので古いのと、加えて、OWASP Top 10以外には、(少なくとも調べた範囲では)スマホクライアント固有の問題をわかりやすくまとめたオフィシャルな情報源は無かったため、今回はセキュリティ室側で作成した資料ベースで話をしていこうと思います。
主な題材としては、ゲーム運営をしている当社としては知っておいて欲しい「スマホゲームのチート」になります。
これを主なテーマに、攻撃の手法やリスクなどをお話したいと思います。

チートについてのアジェンダ

MIXI
CONFIDENTIAL

- チートとは
- チーターの心理
- チートのリスク
- チート手法の紹介
- チートと法律
- まとめ

チートとは

- チート（英: cheat）とは騙す、欺くこと。
- **コンピュータゲーム**において、広義には制作者が**意図しない方法**や結果により使用者が意図的に**公平性を損なわせる**行為のこと。
- 狭義には、コンピュータゲームにおいて優位に進めるための（バグ等を用いた）不正行為またはハッキング行為のこと。
- 製作者が意図して組み込み・公開した機能は仕様でありチート行為とは称されないが、非公開機能や仕様バグ、実装バグを用いた「バグ技」はその名の通りチート行為である。

引用元：https://ja.wikipedia.org/wiki/%E3%83%81%E3%83%BC%E3%83%88, (2022/4/26)

MIXI

実際にチートの様子を見てみましょう

チートの例

MIXI
CONFIDENTIAL

- アイテムの増殖
- ステージの全開放
- 獲得アイテムの改変
- お金無限
- ステータスの改変（敵・味方）
- 当り判定の拡大や縮小
- スコアの改変
- 壁が透ける
- 空を飛ぶ
- 位置情報を偽装する
- ....etc

ゲームによって、ズルの仕方はさまざま。

チーターの心理

- チーターの心理には、大きく3種類があるように見受けられる。
  - ラクして結果を出したい
    - 一番自然であろう心理。ラクに強くなって気持ちよくなりたい。
  - 「チートしちゃってるオレ」感を味わいたい
    - チートできる/しているということ自体に快感。
      - Mod(改造したアプリ)をネットに公開
      - チートのやり方をYoutube等で解説
      - ↑「ラクして強くなりたい」だけならそれを世間に知らせるようなことはしないはず。
  - 金銭を得たい
    - チート代行業で稼ぐ
    - チートで得たアイテム/アカウントを販売で稼ぐ
    - チート情報をまとめたサイトの運営で稼ぐ

チーターを取り巻くWin-Winな環境

ラクして強くなれて嬉しい

ラクして強くなりたい人

代行依頼/アカウント購入

お金が手に入って嬉しい

金銭を得たい人

チート代行業で稼ぐ
チートで得たアイテム /アカウントを販売で稼ぐ
チート情報をまとめたサイトの運営で稼ぐ

※詐欺も行っているかもしれない。

情報を得て自らチート

商売ネタの仕入れ

自己顕示欲が満たせて嬉しい

チートしちゃってるオレ

チートアプリや
手法を公開

・改造アプリ
・チート方法の解説

私たち運営側にとってのチートのリスク

MIXI
CONFIDENTIAL

- ゲームプレイにおいて不公平が生じてしまう。フェアでない。
  - スポーツでいえばドーピング。
  - ユーザは冷める。そして運営に対するヘイトが溜まる。
- ゲームの寿命の短縮。
  - すぐクリアされるという直接的な側面と人離れによる間接的な側面。
- 課金数の低下、課金機会の損失。
  - チートしている人→チートした方がコスパ良い
  - チートしていない人→馬鹿らしくなって課金しなくなる
- 関係会社への被害
  - 例えばコラボ先の商品を買うとそのIPのキャラデータが手に入るキャンペーンがあったとして、不正にキャラデータを入手できるチートが行われた場合、コラボ先商品の売り上げが下がり、関係会社に対する被害につながる。
- ブランドイメージの低下。
- ユーザ対応コストの増加。
  - 通報対応/BAN対応/問い合わせ対応など。

チートの手法紹介

MIXI
CONFIDENTIAL

チートの手法を下記４種類に大別して「どうやってやるか」「どう対策するか」をご紹介。

- メモリ改変
- 通信改変
- ローカルデータ改変
- クライアントアプリ解析・改造（リバースエンジニアリング）

メモリ改変

プロセスメモリエディタと呼ばれるツールを使用して、メモリに格納される値（HP・攻撃・お金など）を改変する。基本的にはjailbreak・root化が必要(不要なものもある模様)。

【基本的な使い方】

1. ターゲットとなるプロセスを選択する。
2. 書き換えたい値（HP・攻撃・お金など）をメモリ内から検索する。
   ※多数検索ヒットする場合は値を変化させて絞込み検索を行う。
3. 目当ての値のメモリ番地が特定できたら、好きな値に書き換える。

プロセスメモリエディタの例

- iGameGuadian
- GameGuardian
- GameGem
- GameKiller
- CheatEngine　※PC

メモリ改変の例

現在の所持金は1000G

※一般的なRPGの買い物画面だと思ってください。
ここからプロセスメモリエディタを使用して、
所持金1000Gを増やす場合の流れを説明します。

メモリ改変の例


Lvl 99
いらっしゃい
買うぞ
shop
shop
所持金
1000G
買う
売る
どうぐ
はなす
そせいやく 500 G x 1
やくそう 50 G x 0
どくけし 999 G x 0
フリスク 100 G x 0
現在の所持金は1000G
プロセスメモリエディタ
で「1000」を検索
Lvl 99
1000
検索
7件 ヒット
000D7DD0 1000
000D7DD4 1000
000D7DD8 1000
000D7DDC 1000
000D7DE0 1000
000D7DE4 1000
000D7DE8 1000

メモリ改変の例


Lvl 99
いらっしゃい
買うぞ
shop
shop
所持金
1000G
買う
売る
どうぐ
はなす
そせいやく 500 G x 1
やくそう 50 G x 0
どくけし 999 G x 0
フリスク 100 G x 0
現在の所持金は1000G
プロセスメモリエディタ
で「1000」を検索
1000で検索すると 7件
Lvl 99
1000
検索
7件 ヒット
000D7DD0 1000
000D7DD4 1000
000D7DD8 1000
000D7DDC 1000
000D7DE0 1000
000D7DE4 1000
000D7DE8 1000
どれが所持金かわからない

メモリ改変の例

1つ500Gの「そせいやく」を購入します。

現在の所持金は500G

先ほどの7件の中から目当ての値をみつけるために、
「500」で絞込み検索を行います。

メモリ改変の例

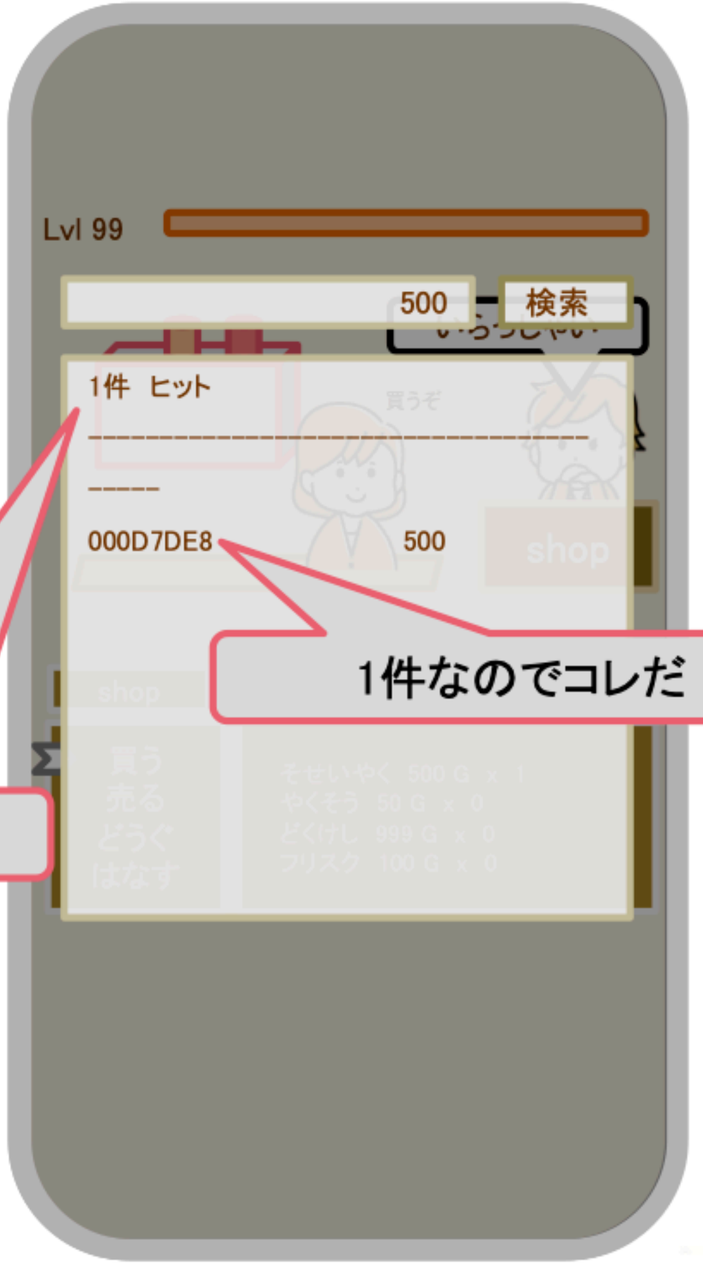

Lvl 99
そせいやく
1つ 買います
shop
うい
shop
所持金
500G
買う
売る
どうぐ
はなす
そせいやく 500 G x 1
やくそう 50 G x 0
どくけし 999 G x 0
フリスク 100 G x 0
現在の所持金は500G
プロセスメモリエディタ
で「500」を検索
Lvl 99
500
検索
1件 ヒット
000D7DE8
500

メモリ改変の例

現在の所持金は500G

プロセスメモリエディタ
で「500」を検索

500で検索すると1件！



1件なのでコレだ！

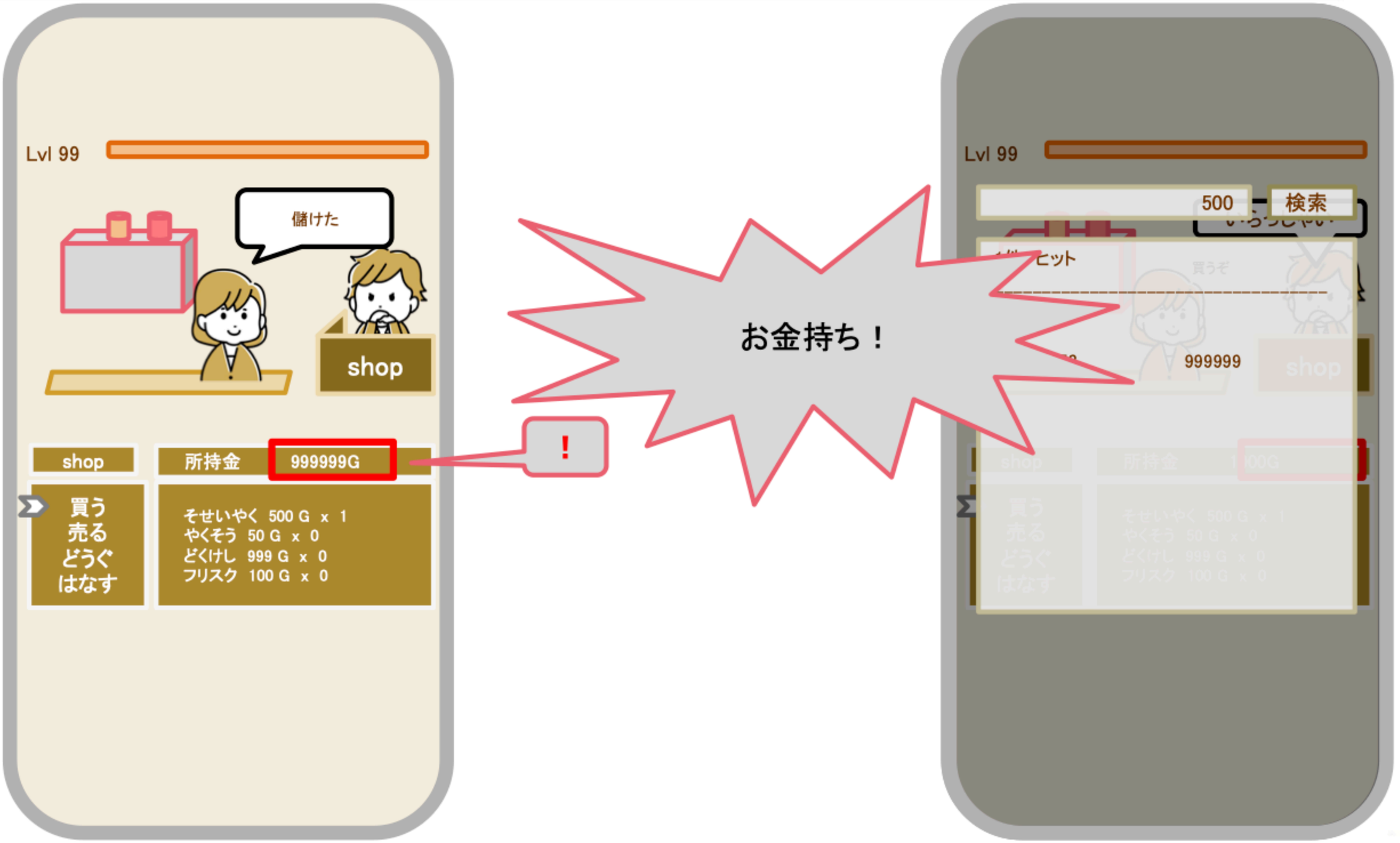
メモリ改変の例

所持金を保持している箇所が判明したので、
好きな金額に改変します。

500 を 999999 に改変

この状態でゲームに戻り、表示が更新されると…

メモリ改変の例


Lvl 99
儲けた
shop
shop
所持金
999999G
買う
売る
どうぐ
はなす
そせいやく 500 G x 1
やくそう 50 G x 0
どくけし 999 G x 0
フリスク 100 G x 0
!
お金持ち！
Lvl 99
500
検索
999999
shop

メモリ改変の対策

MIXI
CONFIDENTIAL

- サーバ側
  - 重要な値の保持、重要な処理はサーバ側で行う。
  - 先の例で言えば、お金の値と、「買う」という行為の演算処理はサーバ側で行うようにする。

- クライアント側（保険的）
  - 簡単に検索できない形に変えて保存しておく。
  - 「100」という値を保存する場合、100そのままを保存するのではなく、
    - 保存時→100+12345=12445
    - 参照時→12445-12345=100
  - というように、何らかの変形処理を施したうえで保存しておく。こうすることで、チーターからすると値が検索できなくなる。
    - 上記は一例であり、「検索できない形」にする処理なら何でもOK

通信改変

MIXI
CONFIDENTIAL

ローカルプロキシツールなどを利用し、クライアントとサーバの間の通信を改変する手法。

チートの例

- ゲームのプレイ結果をサーバに送信する際、スコアを書換える。
- キャラステータスをサーバから受信する際、攻撃力を書換える。

ローカルプロキシツールの例

- Fiddler
- Burp Suite　←ハンズオンで使った！

通信改変

MIXI
CONFIDENTIAL

通信改変がどう行われるかの図

先ほどのハンズオンでやったことのゲーム版！
ハンズオンではスマホアプリの代わりにブラウザ、
ゲームサーバの代わりにJuiceShopでしたが、基本的な方式は同じ！

通信改変の例

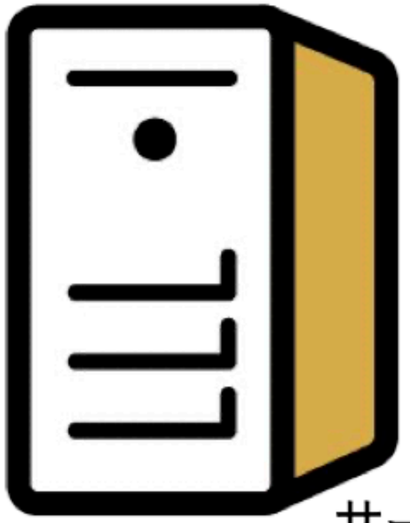
Now
Loading...

ステージ1
近所の草むら

ゲームをしていると、このような場面を
見かけることがあると思います。

「ステージ1」の情報を下さい

「ステージ1」のデータどうぞ

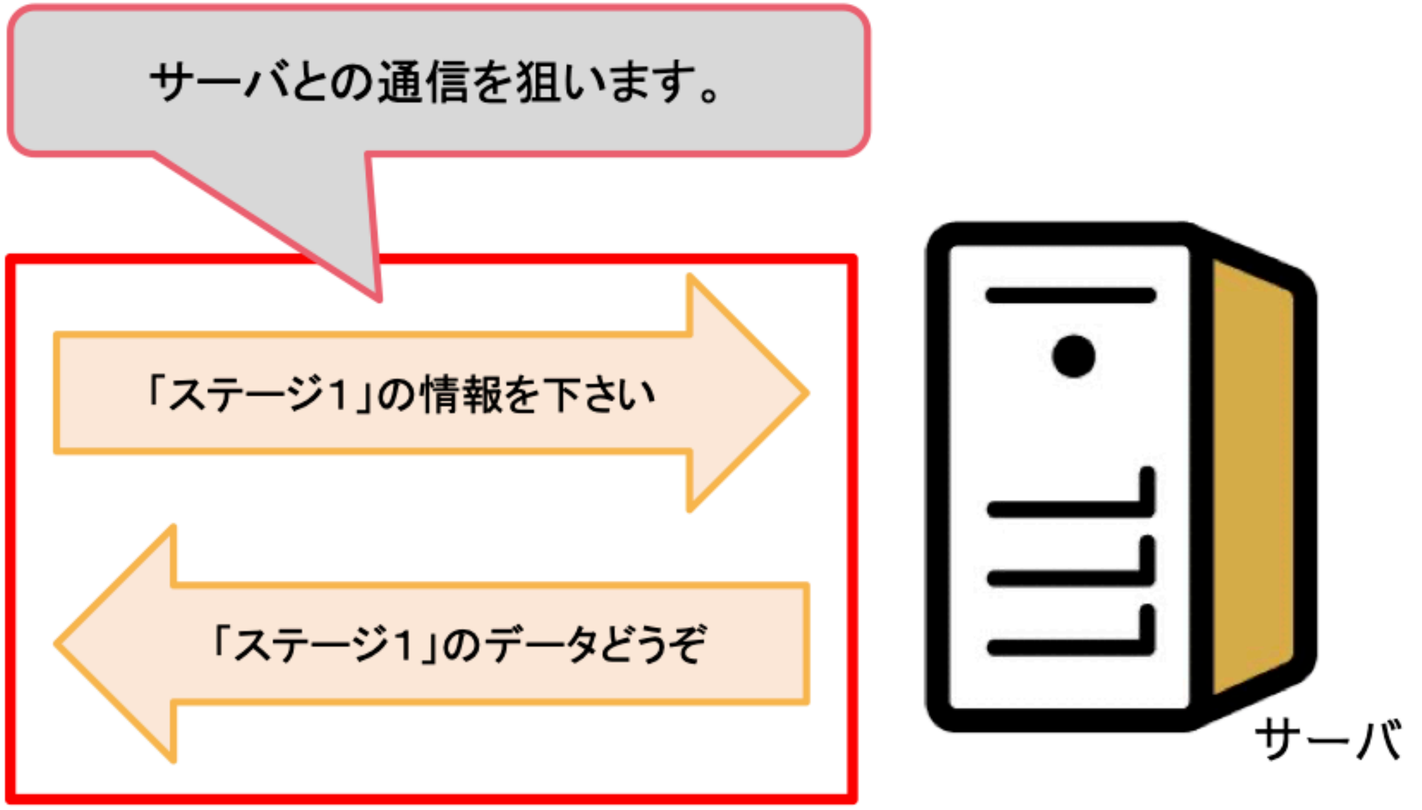


サーバ

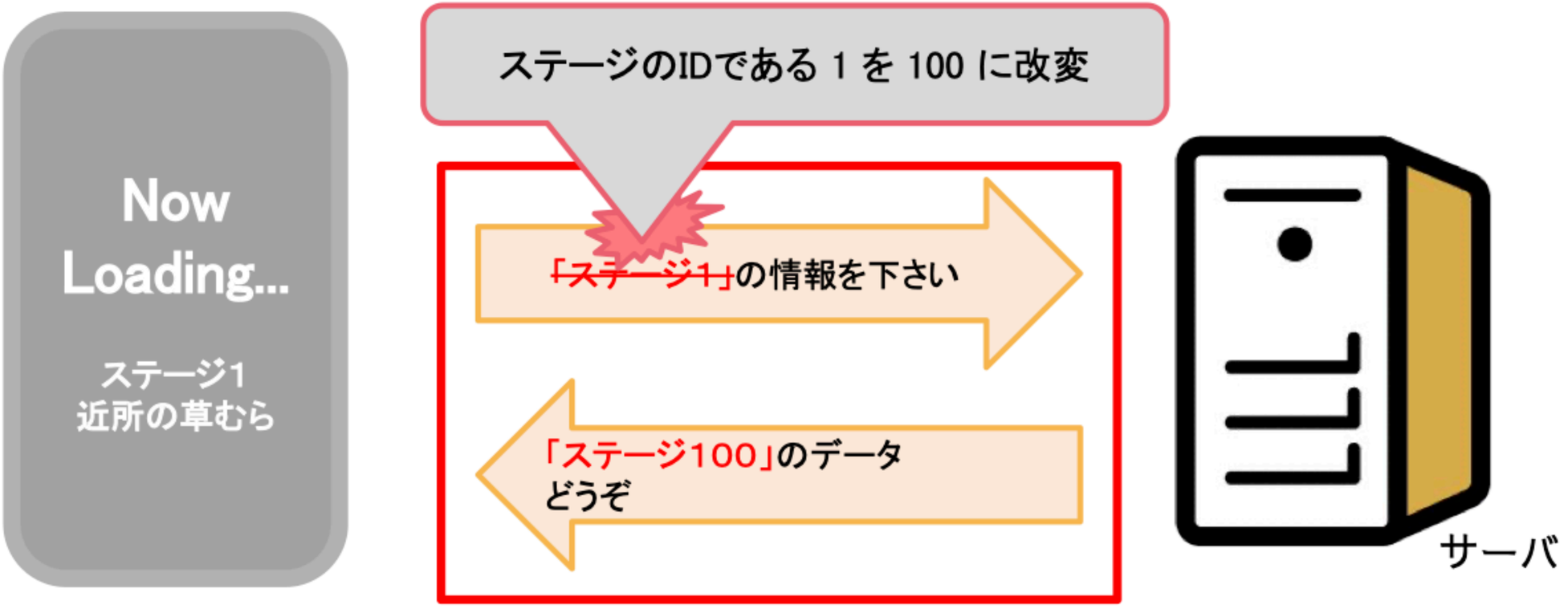
通信改変の例

Now
Loading...

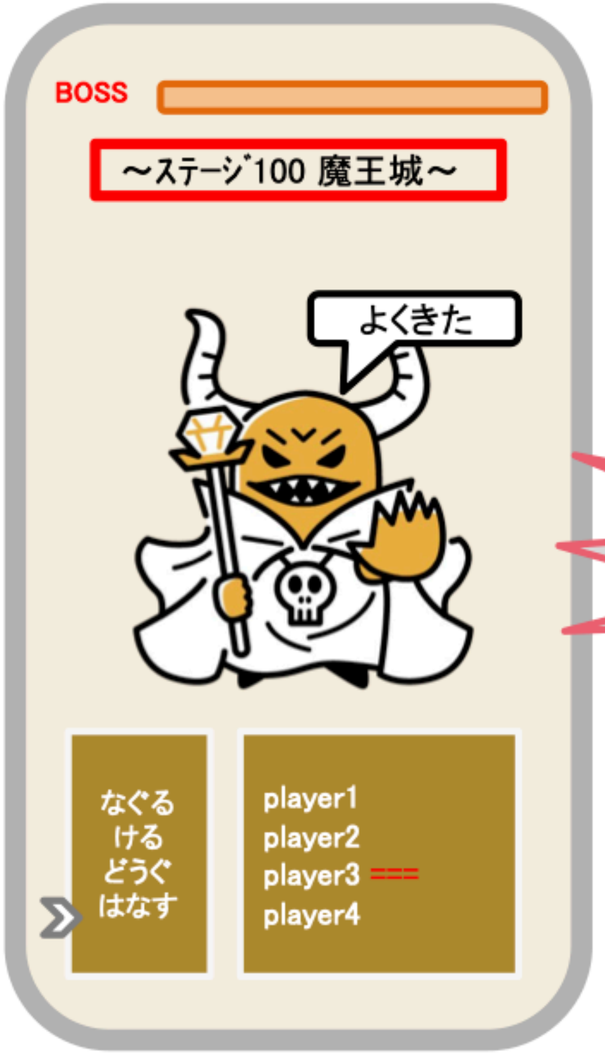
ステージ1
近所の草むら



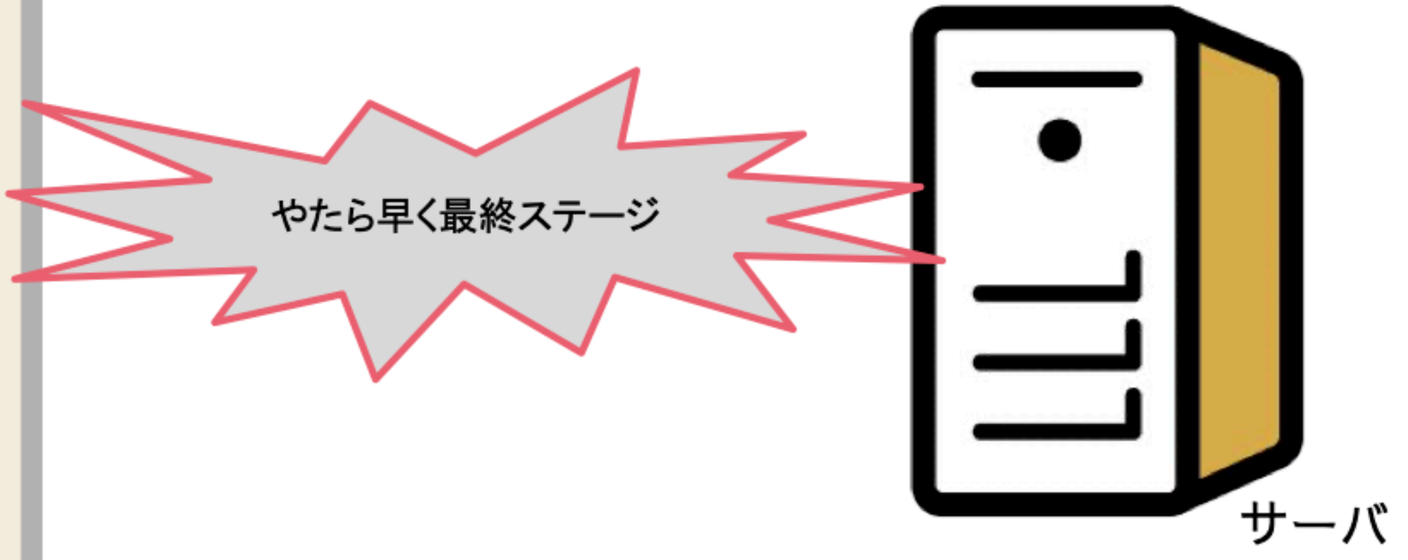
通信改変の例



通信改変の例



成功すれば、いきなり最終ステージに
チャレンジできる場合などがあります。



通信改変の根本的対策

MIXI
CONFIDENTIAL

- ユーザに改変されると困る処理はパラメータに持たせない。
  - ガチャを引く際、消費オーブ数をパラメータで渡している場合
    → オーブ数をパラメータに持たせるのではなく、ガチャIDだけ送るようにする。
    そうすると消費オーブ数を改変できなくなる。
- 持たせている場合は正当性を検証する。
  - 先の例なら、「このユーザはステージ100に挑戦できる進行度合いか？」をサーバ側で検証する。

通信改変の保険的対策

MIXI
CONFIDENTIAL

- 以下を併せて実施する。
  - パラメータとともにパラメータのハッシュ値も送信するようにし、改ざんチェックを行う。
  - SSL/TLSを利用するとして、それに加え
    - アプリケーション層でパラメータ全体を暗号化する。
    - SSL/TLS証明書のピンニングを行う。

通信改変

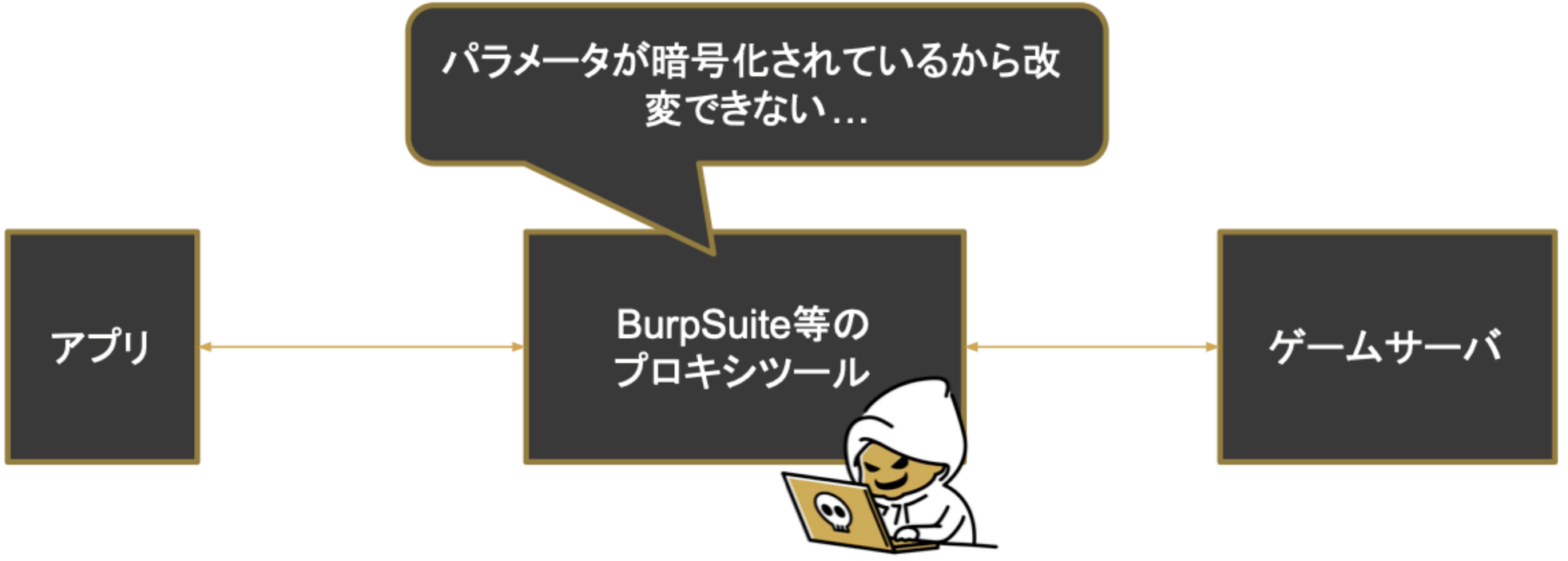
MIXI
CONFIDENTIAL

通信改変がどう行われるかの図

先ほどのハンズオンでやったことのゲーム版！
ハンズオンではスマホアプリの代わりにブラウザ、
ゲームサーバの代わりに Juice Shopでしたが、基本的な方式は同じ！

通信改変

パラメータ全体をアプリケーション層での暗号化をする。



通信改変

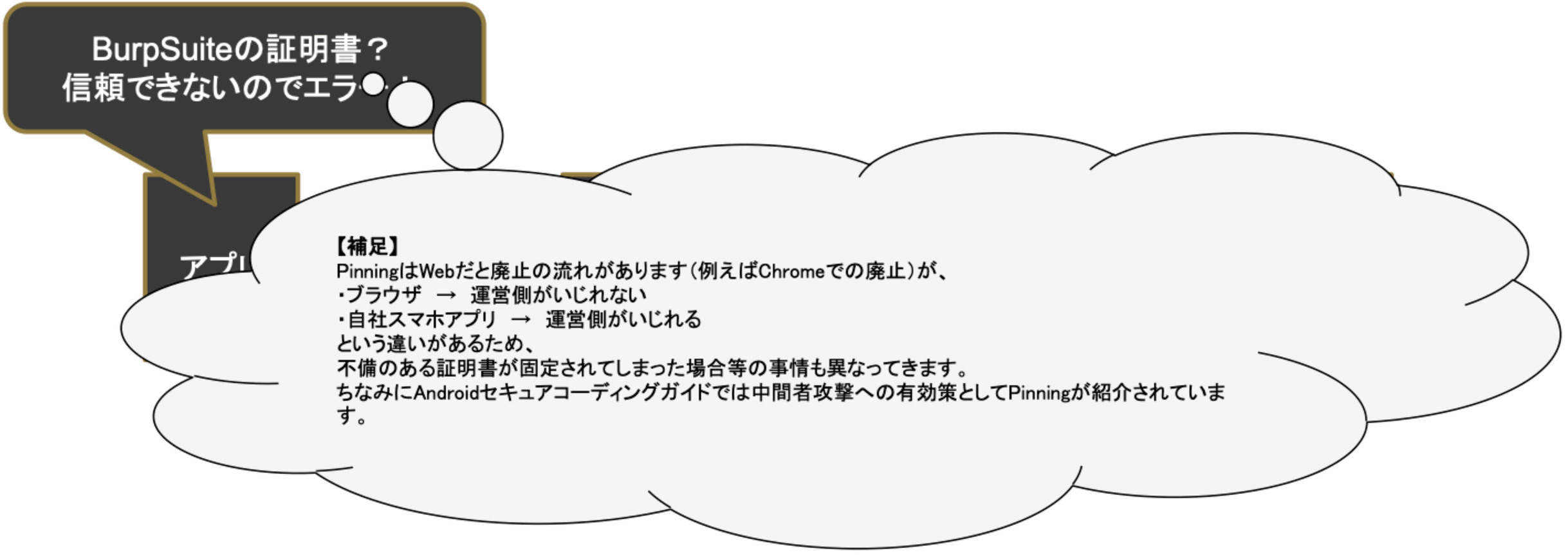
MIXI
CONFIDENTIAL

SSL/TLS証明書のPinningを行う。

通信改変

MIXI
CONFIDENTIAL

SSL/TLS証明書のPinningを行う。



ローカルデータ改変

ローカルストレージやSDカードなど、ファイルにセーブデータやゲームに関する情報を保存している場合、その値を改変し、チートが可能な場合がある。

例

- ゲームのセーブデータをファイルに保存する設計だったとして
- ファイル内にあるレベル/hp/atkに相当するパラメータを書き換えることで
- ゲーム再起動時、そのファイルに記載されたレベル/hp/atkに強化されている

ローカルデータ改変の対策

MIXI
CONFIDENTIAL

根本的対策

- ゲームのセーブデータなど重要な情報はローカルに保存しない。サーバー側に保存する。

保険的対策

- やむを得ずローカルに保存する場合は暗号化をし、改変の難度を上げる。

クライアントアプリ解析・改造

アプリ自体の改変。
アプリを下記のようなツールを用いて逆コンパイル/逆アセンブルし、内容を解析、改ざんする手法。
行うにはプログラムのロジックを理解する/改造するための技術力が必要になる。

解析に用いられるツールの例

- JD-GUI　（逆コンパイラ　Java）
- ILSpy　（逆コンパイラ　C#など）
- IDA Pro　（逆アセンブラ）
- Ghidra　（逆アセンブラ）

クライアントアプリ解析・改造

「行うにはプログラムのロジックを理解する/改造するための技術的力が必要になる」なら、悪用は難しい？

→改造をするのは難しいかもしれない。
　ただし、改造後のアプリを使ってチートすることは簡単。

- 改造アプリは容易に入手可能
- 改造アプリをインストールしてしまえばチート自体は容易

クライアントアプリ解析・改造の対策

MIXI
CONFIDENTIAL

解析・改造行為そのものを完全に防ぐことはできない。
なるべく解析しづらく、改造しづらくするしかない。

- コードの難読化
- アプリの改ざんチェック
  - （改ざんチェックを無効化するように改変することも可能なため、根本的対策にはならないが難度を上げるという意味ではやった方がセキュア。）

マクロ

ゲームの処理(レベル上げ等単純な処理)を自動化し、ゲーム内で利益を得ようとする手法。
一応、ゲームに対して行っていること自体は正規の行動のため厳密にはチートではないが、
「ラクをして強くなる」という点においては近い行為。
スマホゲームにおいて行われるマクロの方法はいろいろある。

- エミュレータでアプリを動作させ、PCの操作記録ツールを使用する。
- マクロ用のスマホアプリを使用する。
- **実際のゲームフローで送信される通信リクエストを再現/送信することで、時短ゲームクリア等を行うパターンもある。**
  （これはもはやマクロというより通信改変の亜種かも）。

エミュレータの例
- Genymotion
- BlueStacks
- Andy
- NoxPlayer

マクロを組むツールの例
- Auto Clicker(スマホアプリ)
- ステップ記録ツール(Windows)
- Red Magic(マクロ機能内蔵スマホ)

マクロ

MIXI
CONFIDENTIAL

ゲームの処理(レベル上げ等単純な処理)を自動化し、ゲーム内で利益を得ようとする手法。
一応、ゲームに対して行っていること自体は正規の行動のため厳密にはチートではないが、
「ラクをして強くなる」という点においては近い行為。
スマホゲームにおいて行われるマクロの方法はいろいろある。

- エミュレータでアプリを動作させ、PCの操作記録ツールを使用する。
- マクロ用のスマホアプリを使用する。
- **実際のゲームフローで送信される通信リクエストを再現/送信することで、時短ゲームクリア等を行うパターンもある。**
  （これはもはやマクロというより通信改変の亜種かも。）

エミュレータの例

- Genymotion
- BlueStacks
- Andy
- NoxPlayer

マクロを組むツールの例

- Auto Clicker(スマホアプリ)
- ステップ記録ツール(Windows)
- Red Magic(マクロ機能内蔵スマホ)

ゲーム内容次第では、
ゲームバランスを壊す可能性がある。

マクロの緩和策

MIXI
CONFIDENTIAL

- 通常プレイではあり得ない値(スコア/クリア時間等)ではないかサーバ側でチェックする。
  - その処理に対しては報酬を与えない
    - たとえばスロットの場合、スロット回転が終わるまでの最低時間を予め計測しておき、それよりも明らかに早い間隔で回されている場合はお金だけ減らして報酬を与えないようにする。
  - ログをとっておき、BANする
    - たとえばクエストの場合、明らかに規定クリア時間より早いものを見つけたら記録しておく。後からBANする。
- 自動化のための通信リクエスト再現を難しくする。
  - リクエストパラメータの暗号化
  - リクエストの改ざん検知
  - コードの難読化

チートと法律

MIXI
CONFIDENTIAL

Q. チートって違法なの？

A. 違法(犯罪)になる可能性はあります。

Confidential

Confidential

事例から学ぶポイント

MIXI
CONFIDENTIAL

- チートは刑事事件になりうる
- チートが該当しうる罪状は複数種類ある
  - 私電磁的記録不正作出・同供用
  - 不正競争防止法違反
  - 電子計算機損壊等業務妨害
  - 著作者人格権侵害
  - Etc

チートそのものを単独で指した罪状があるわけではなく、
様々な法的観点に違反する可能性がある。

チートと対策はイタチごっこ

- サーバ側はセキュアに守れる場所である一方、クライアント側は「難しくする」という対応しか取れない。完全には守れない。
- 一方で、バトル等のアクション性の高い部分は現状クライアントに置かざるを得ない場合が多い。

- 結果、
  - チーターがチートをする
  - 運営が対策をする
  - チーターがそれを回避してチートする
  - 運営がまた対策する
  - ,,,

  というループが起こり得るので、イタチごっこと言われがち。

一体どうすれば...

MIXI
CONFIDENTIAL

現状

- ガチャ等の絶対チートされちゃダメな値/処理は必ずサーバ側に寄せる。
- そうでないものは現実的には、「ここまでは検証する」「後からBANで対応する」など、ケースバイケースで柔軟に対応していくしかない。

未来

- クラウドゲーミングが主流となり、ゲーム処理が丸ごとクライアントから手の届かない所で行われるようになればチートの激減した世界がやってくるのかもしれない。

チート以外で気をつけるべきこと

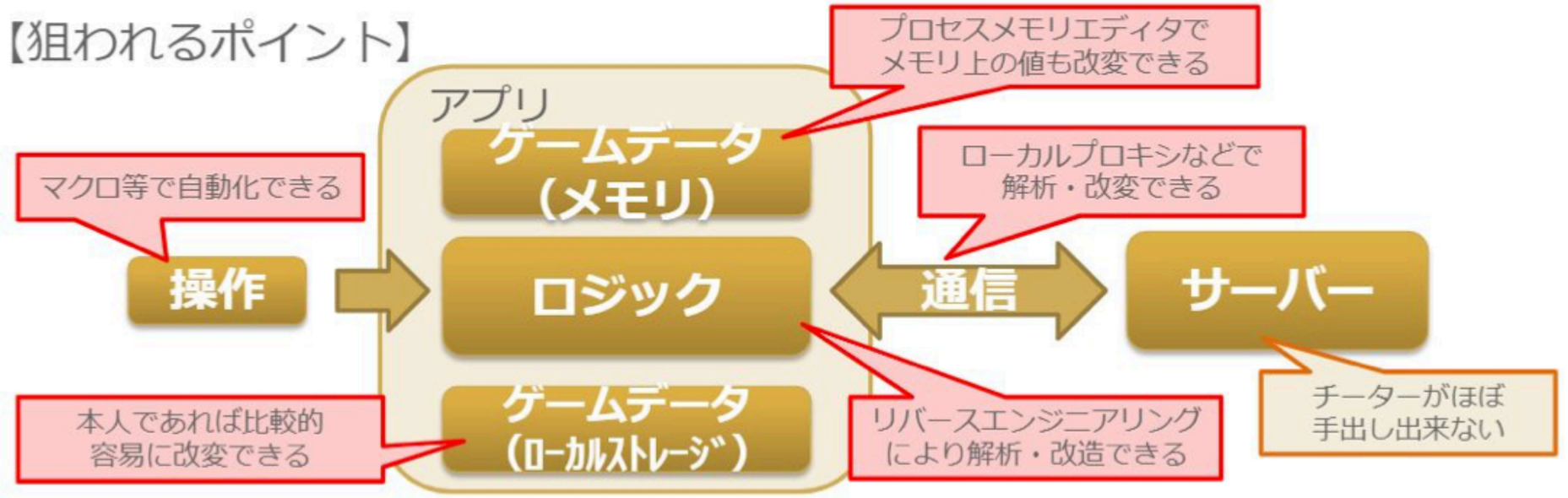
MIXI
CONFIDENTIAL

- ログに機微な情報が載らないように注意。
- SDカード等の、他アプリからアクセスできる領域に重要なデータを保存しない。
- カスタムURLスキームでは重要なデータを取り扱わない。
  - 同じカスタムURLスキームを宣言した偽装アプリにデータが渡ってしまうかもしれない。
- レシート検証はちゃんとしましょう。改ざん/別レシート/過去レシートなど。
  - レシート検証周りの参考
    - [iOS]https://developer.apple.com/jp/documentation/storekit/in-app_purchase/validating_receipts_with_the_app_store/
    - [Android]https://developer.android.com/google/play/billing/security

チートならびにクライアント側のセキュリティのまとめ

MIXI
CONFIDENTIAL

- 下記がチートで狙われるポイント。図からも分かる通り、クライアント側のデータ/処理を守り切ることは、難しい。
- そのため、何度も言っていますが重要なロジックや情報はサーバ側に寄せて守りましょう。



- どうしてもサーバ側に寄せられないものについての対応は、ケースバイケースとなります。サービスの性質/改修コスト等に応じて、適切な緩和策を考えていく必要があります。
- ゲームのチートでも、そうでなくても、この考え方は基本であり、同じです。

参考ドキュメント

MIXI
CONFIDENTIAL

- セキュアコーディングガイド
  - Android (具体的なコードを交えて実践的に書かれているのでお勧め)
    - http://www.jssec.org/dl/android_securecoding.pdf
  - Apple (iOSの具体的な話というよりは一般論的な話)
    - https://developer.apple.com/library/content/documentation/Security/Conceptual/SecureCodingGuide/Introduction.html
- OWASP Mobile Top 10
  - https://owasp.org/www-project-mobile-top-10/

MIXI

いきなり全部は難しいので...

MIXI

まず、攻撃やリスクの存在を認識しましょう！

MIXI

自分の行動や処理の結果に想像力を働かせよう！

MIXI

あぶないかも？　と思ったら周りに相談しよう！

困ったときの相談先

MIXI
CONFIDENTIAL

Confidential

心もつなごう。MIXI