[めもおきば てっくれぽーと] クラウドとセキュリティ TL 流速を支えるエンジニアの総合誌

めもおきば Tech Report 総集編

Cloud Security
Serverless IoT

Hardening SecCamp
SORACOM Bluetooth
Realtime

Vol.1

2015.12 - 2018.10

めもおきば TechReport
総集編 Vol.1

[2015.12 - 2018.10]

Aki @ nekoruri

はじめに

本誌は、2018 年までにサークル「めもおきば」から頒布された「めもおきば TechReport」から、「サーバーレスの薄い本」シリーズを除いた以下の号の再編集版です。テーマ別に整理のうえで加筆や補足を加えたものです。

- TechReport 2015.12（コミックマーケット 89）
  - Hardening 10 ValueChain 体験記
  - クラウド四方山話「サーバーレスアーキテクチャ」
  - 2015 年振り返りと 2016 年予想
- TechReport 2016.12（コミックマーケット 91）
  - 2016 年振り返りと 2017 年予想
- TechReport 2017.08（コミックマーケット 92）
  - SORACOM 買収からわかる IoT の未来
  - 私もセキュリティ・キャンプに参加したい！
  - Bluetooth mesh ってどうよ
- TechReport 2017.12（コミックマーケット 93）
  - Hardening 2017 Fes レポート
  - 2017 年振り返りと 2018 年予想
- TechReport 2018.04（技術書典 4）
  - なろう！フルスタックエンジニア
  - htc vive 買ってみた
  - Microsoft MVP Global Summit 行ってきた
- TechReport 2018.10（技術書典 5）
  - サーバーレスで実現するストリーム処理徹底活用術

目次

Hardening 10 ValueChain 体験記

TechReport 2015.12

Hardening Project という競技をご存知ですか。

技術用語として Hardening（ハードニング）とは、「脆弱性を減らすことでシステムのセキュリティ堅牢にすること」（Wikipedia）を指します。サービスを提供するエンジニアであれば誰でもやっている当たり前のことですね。この Hardening――すなわち「守る技術」を競うため、WASForum[1]が 2012 年から実施しているセキュリティ・イベントが Hardening Project です。

以前から Hardening Project の存在は知っていたのですが、某ゆるふわ勉強会 [2]の人たちに誘われて、今年 2015 年 11 月に行われた Hardening 10 ValueChain（以下「h10v」）で初参加してきました。結果としては最下位という惨敗でしたが、とても価値のある経験を得られたので少しでもそれが伝わればと思います。

どんなイベント？

ざっくりと言えば、各チームは「株式会社はーどにんぐ」の一員となり、脆弱性のある EC サイト環境を競技時間のあいだにできる限り堅牢にして外部からの攻撃に耐え、最終的に競技終了時点での売り上げを競うという競技形式のイベントです。ただ、こんな感じのふわっとした理解のまま参加すると、あっさり負けます（ソースは自分）。

今回の h10v の公式サイトには以下のように書かれています。

> 防御技術力のみならず、サイトの運営を維持する総合力、つまり攻撃に対する堅牢化、それと同時に売り上げの確保、ダウンタイムの最小化、そのためのコミュニケーションスキルのすべてを加味して勝者が決まる [3]

このような評価軸を実現するため、ここに書かれた様々な「総合力」が無ければ「売上」というスコアが上がらないように、競技全体が設計されています。

---

[1] Web Application Security Forum http://wasforum.jp/

2 #ssmjp http://ssm.pkan.org/

[3] http://wasforum.jp/hardening-project/

競技そのものが「Hardening Day」として丸一日掛けて行われますが、さらにその振り返りとして「Softening Day」が翌日にあります。各チームの取り組みを発表し、攻撃側からの答え合わせを経て表彰式が行われます。Softening Day の内容は YouTube にて公開されています [4]。

それでは、実際の h10v における競技の詳細を紹介します。

全体像

h10v では、10 人が 1 チームとなり、全 6 チームで競います。チームごとに用意されたテーブルに集まって 8 時間を戦い抜きます。人数や競技時間は、各回のテーマに応じて変わることがあるようです。前回（Hardening 10 MarketPlace）では 1 チーム 5-6 人だったようです。

各チームには、複数の仮想サーバで構成される競技環境が用意されます。各テーブルに用意された有線 LAN を経由して競技時間中のみ競技環境に接続できます。詳しくは後ほど詳しく紹介しますが、EC サイトをはじめとするいくつかのシステムが用意されています。この EC サイトに対して、運営側の用意したクローラーが「お客様」として定期的に訪問して商品を買っていきます。この「お客様」による売上金額こそが、スコアとして順位付けの対象となります。あくまで競技としては、システムの堅牢化はあくまで手段に過ぎず、購入数を上げることが目的となります。

この購入数を決めるのが「繁盛レベル」です。繁盛レベルは、運営側（後に紹介する kuromame6）が規定のルールに従って操作しているパラメータで、リアルタイムで会場前面のスクリーンに映し出されます。この繁盛レベルは「評判」を模しているため、たとえば情報漏洩事件があると一気に叩き落とされ、適切な対応をすれば回復してきます。また、事件そのものを未然に防ぐことに成功すれば上がるようです。上手く社長や顧客などへの技術アピールに成功すればさらなる向上が見込めます。これが「システムの堅牢化と安定運用」を「売上」として評価するための枠組みの肝です。

堅牢化したはずのシステムに対して「外部からの攻撃」を行うのが、運営側の「kuromame6」と呼ばれる技術チームです。彼らは 8 時間という競技時間の間、あの手この手でシステムの脆弱性を突いてきます。実際に行われた攻撃について後ほど紹介します。kuromame6 は攻撃だけでなく、競技環境の構築・運用や評価なども担当します。

「段取り八分」という言葉があるように事前準備も重要です。今回は「参加者配付資料」が前日の夕方に配布されました。それだけでなく、そもそも h10v の開催告知からも、前回（Hardening 10 Marketplace）の発展であることが読み取れるため、競技環境の大枠そのものは大きく変わっていないだろうということが予想できます。したがって、これらの事前情報を基にした準備をすることで、8 時間という短い時間を何十何百時間にも拡張することができます。

[4] https://youtu.be/I1Um1l_9Vw0　https://youtu.be/nWPwbCbktIY

今回は 10 人という大きなチーム（前回は 5-6 人）ですので、チームの体制作りやタスクマネジメントも重要です。募集要項 [5]においても明言されています。

> すべてのチームは「やることが多すぎて手がでなかった」ということは激減し、むしろ「チームワークで乗り越えられた！」という経験ができる可能性が飛躍的に向上することでしょう。

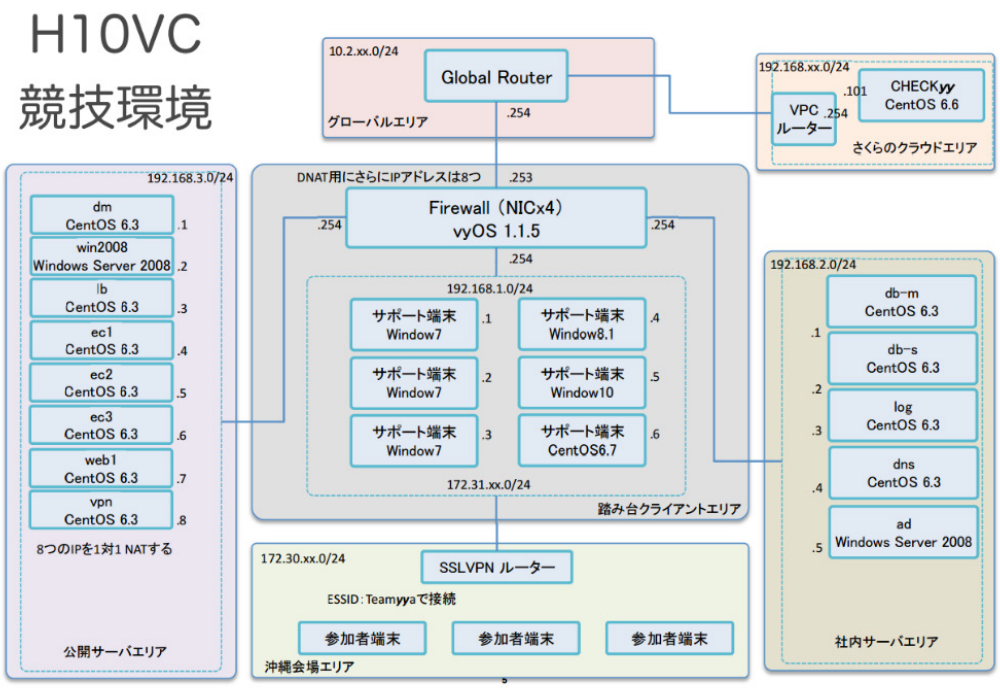
人数が多いだけでなく、自分のチームに足りない部分を補うための「マーケットプレイス」もまた用意されています。マーケットプレイスにお金を落とすことで、カスペルスキーや NEC の WAF などのソフトウェアを購入したり、プロのセキュリティエンジニアを時間で契約したりといったことができます。

これが h10v という競技の全体像です。上手く機能するチームを組み立てて事前準備を進め、攻撃側の kuromame6 に先んじて渡されたシステムを素早く堅牢化し、もし脆弱性を突かれたなら適切に対応するというだけでなく、繁盛レベルの増減条件を見極めて積極的に EC サイトの価値を上げていく——ここまでやって始めて勝利できる、そんな感じの総合競技です。

[5] http://wasforum.jp/hardening-project/h10v-application/

サーバ環境

以下が h10v で各チームに提供される競技環境です [6]。



12

各チームは大きく分けて 4 つのネットワークのサーバ・クライアントを管理します。これらはグローバルエリアのルータを介して、仮想的な「インターネット」に接続されています。kuromame6 からの攻撃や、EC サイトへのクローラーはグローバルエリアを介して「インターネット」側からやってきます。

「社長」や「お客様」などと、「インターネット」を介してメールをやりとりすることも可能です。ただし、現実のインターネットへの接続は制限されているため、外部サイトから直接ファイルを持ってくることができないことには注意が必要です。

[6] http://www.iij.ad.jp/company/development/tech/techweek/pdf/151112_2.pdf

■ 公開サーバエリア

公開サーバエリアには、EC サイト本体や「株式会社はーどにんぐ」のコーポレートサイトなどを提供するサーバ群が設置されています。また、外部との連絡を行うための DNS 兼メールサーバもあります。

「社長」などへの適切な報告がされないと繁盛レベルは上がらないため、EC サイトなどのウェブサーバだけを守っていれば良いわけでは無く、実際にこの DNS サーバに対しても攻撃が行われました。

■ 社内サーバエリア

データベースサーバや、Active Directory サーバなどが置かれています。DB サーバが落とされれば当然ながら公開サーバエリアで提供している様々なサービも停止しますし、Active Directory サーバが被害を受けると、この後に紹介する踏み台クライアントにも影響します。

■ 踏み台クライアントエリア

公開サーバと並んで重要なのが、ここに設置されたサポート端末です。h10v では参加者の端末から直接接続して良いのはサポート端末のみというルールがあります。したがって、サポート端末が被害を受けてしまうと公開サーバなどに対して操作ができなくなってしまいます。

唯一 SSH が可能で事実上の作業機点となる CentOS 端末は 1 台しかありませんし、5 台の Windows 端末にはそれぞれメールが設定され、ここでしかメールの送受信ができません。しつこいようですが h10v では報告こそが評価の根幹ですので、CentOS 端末だけでなく Windows 端末もまた、EC サイトと同じかそれ以上に死ぬ気で守る必要があります。

■ さくらのクラウドエリア

ファイアウォールの外から「インターネット」を介して外部から自チームの環境を動作するために、「確認用サーバ」が設置されています。ごく普通の CentOS 環境のサーバが用意されているようなのですが、今回うまく活かすことができませんでしたので詳細は不明です。

実際の攻撃

h10v で実際に行われた代表的な攻撃が以下の通りです。

11 時	標的型攻撃
12 時	業務アプリからの情報漏洩
13 時	DD4BC（DDoS for Bitcoin、30 分間の NW 遮断）
14 時	Malvertising（広告からのマルウェア感染）
15 時	潜んでいたマルウェアの再活動
16 時	応募フォームの Web アプリからの情報漏洩

どれも現実において最近に目立っている攻撃手法です。

これからそれぞれの攻撃について紹介していきます。具体的な攻撃方法の詳細は明かされていないため正確である保証はありませんが、自チームで把握していた内容や、Softening Day などで公表されている内容からの推測をベースにしたものです。

標的型攻撃

前回はメールによる標的型攻撃があったものの誰も「踏んで」くれなかったため、今回は「誰かが踏んだ体でそこからすでに起動していた」というマルウェア感染後という攻撃シナリオになりました。

時間になると「マルウェアに感染して外部に攻撃しているからなんとかしろ」という連絡が外部の組織からやってきます。それと同時に一気に繁盛レベルも引き下げられ、適切な対応が進めば繁盛レベルが回復するという仕組みになっています。

開始から 1 時間での発動ということもあり、これを事前に対策して予防できたチームは無かったように思います。私たちも含めて、カスペルスキーの導入で予防するつもりが間に合わなかったというチームが多いようです。

標的型攻撃で送り込まれたマルウェアは、後ほど再び大活躍します。

業務アプリからの情報漏洩

社長の売上把握や、在庫追加などのバックヤード業務を行うためのウェブアプリが用意されていますが、これがインターネットに全公開されており、そこに表示されていた「最近の購入者情報」に含まれる個人情報が漏洩します。この情報漏洩も「外部からの連絡」として発覚します。

漏洩した情報は以下の組み合わせです。

1. 受注日
2. 購入者の氏名
3. 商品名および単価、数量、金額

参加者配付資料の「社内システム運用ルール」に「情報漏えい発生時の対応」というそのままの項目があるので、基本的にはそれにしたがって対応することになります。

現実でも同じですが、重要なことは、「いつから情報が漏れているか？」をいち早く特定することです。そのためには、この EC サイトのアプリケーションとしての構造を素早く把握しなくてはいけません。また、ユーザへの通知を行うためには、漏えい対象のユーザのメールアドレスを取得してお詫びメールを送るといった運用作業も必要となってきます。だんだんと、総合的なサービスの運用能力が問われるという Hardening Project の本質とが見えてきます。

DD4BC（DDoS for Bitcoin、30 分間の NW 遮断）

13 時に競技環境のネットワークに突然アクセスができなくなります。

DD4BC と呼ばれる「DDoS 止めて欲しければ Bitcoin ちょーだい♡」という攻撃（を模した意図的なネットワーク遮断）です。ネットワークが遮断されてすぐ、kuromame6 から DDoS を受けた旨の共有と、30 分という復旧見込み時間がアナウンスされます。

この 30 分という「何もできない時間」を有効活用し、これまで 3 時間の振り返りと残り 5 時間に向けた体制作りが各チームに求められています。休み時間と思ってカレーをのんびり食べている場合ではありません。

Malvertising（広告からのマルウェア感染）

2015 年後半で最も話題になったマルウェア感染手法である Malvertising（Web 広告からの感染手法）ですが、h10v でも当然のように仕掛けられました。コーポレートサイトに貼られた広告で 3 回に 1 回だけマルウェアが落ちてくるようになっていました（ブラウザ側の脆弱性による drive-by-download 攻撃）。発生時間までに対応ができていなければ、ユーザからの問い合わせという形で発覚します。

教科書的な対応は、広告の管理者に連絡をして、不正な広告が表示されない新しいリンク先 URL を受け取るというものでした。参加者配付資料に「広告サーバ管理者」としてメールアドレスが掲載されていることから、深読みできたチームもあったかも知れません。

ユーザへの対応だけで無く、あわてて状況を確認しようとソフトウェア更新が不十分なサポート端末などでコーポレートサイトを開いてマルウェアに感染してしまえば、その端末が標的型攻撃の経路になりかねません。したがって、発覚後のサーバ側の対応だけでなく、サポート端末側のソフトウェアアップデートや、ウイルス対策ソフト等での防御もまた重要となります。

潜んでいたマルウェアの再活動

標的型攻撃で仕込まれたマルウェアが再び活動を始め、管理者パスワードが Pepper ちゃんにより突然読み上げられたり、Windows 端末を使っていたらマルウェア経由で話しかけられたりなど、潜んでいたマルウェアによる心温まる[7]攻撃がありました。

実際には、Windows 7／Windows 8 サポート端末では脆弱なパスワード・キャッシュから平文パスワードを奪取する攻撃が行われたようです。CentOS 端末ではシェルにキーロガーが仕込まれており、キーロガーの除去もしくは通信の遮断を行わない限り、root パスワード変更などを含むあらゆる操作が kuromame6 に筒抜けになっていたようです。

実際のビジネスが組織で行われることを考えれば、これはとても恐ろしいことです。誰か一人が標的型攻撃の実行ファイルを踏んでさえしまえば、そこを足がかりに最終的には管理権限が完全に奪取されてしまいます。その末路が日本年金機構であったりソニー・ピクチャーズだったりするわけです。強制アクセス制御や、操作端末の制限などの「ぱっと見面倒そう」な最小権限の原則も、こういった実例を目の当たりにすると必要性が実感できました。

正直、もうこのあたりから切羽詰まりすぎて記憶が飛んでいます。つらい。

応募フォームの Web アプリからの情報漏洩

「ありがち」なウェブアプリの SQL インジェクションが最後にやって来ました。Java で組まれたウェブアプリで入力値をバリデーションもエスケープも無く SQL にぶっ込んでしまう素敵フォームです。

このフォーム脆弱性への対応は、かなり各チームで対応が別れたのが面白いところです。ソースコードもサーバ内に同梱されていたので、そもそもの脆弱性自体を直したチームもあれば、フォーム自体を 1 から書き直してデータベースを利用しないメール送信フォームにしてしまったチーム、WAF を入れて対応を保留したチームと様々です。

どれが正解ということはなく、それぞれのチームが持つスキルに合わせた「最適な対応」の判断が求められます。

---

[7] 黙って悪用されて死体すら残らず殺されるよりは余程マシ

結果と反省

我らが Team4「セキュ子の部屋」は残念ながら圧倒的最下位という結果に終わりました。全体の結果は以下の通りです。

順位	見込み販売力	技術点	顧客点	対応点	経済点
1 位: 現地集合検知改ざん	31,142,860	2,450	3,140 [1 位]	3,540 [1 位]	7,300 [1 位]
2 位: ちゃんぷるー	28,799,800	2,150	2,570	2,880	1,300 [3 位]
3 位: SUNSUNSUN	28,367,320	3,200 [1 位]	2,570	2,770	1,100
4 位: ちゅーばしんか	25,991,372	2,750 [3 位]	2,950 [3 位]	3,210	1,600 [2 位]
5 位: 秘密結社にんじんしりしり	25,145,656	2,900 [2 位]	2,570	3,320 [2 位]	1,100
6 位: セキュ子の部屋	14,841,380	2,300	3,140 [1 位]	3,320 [2 位]	500

「見込み販売力」はいわゆるクローラーによる売上金額です。あくまで見込み販売力によって順位は決まりますが、それ以外の 4 つの評価点は以下のように設けられています。

技術点	脆弱性への対応、マルウェアの対応など
対応点	インシデント復旧時間、サイト全体の稼働率など
顧客点	サポート能力、ユーザ保護の取り組み
経済点	サービスや製品をたくさん使って市場を活性化したかなど

私たちのチームは売上では最下位だったものの、外部対応を担当した方の圧倒的努力のおかげで、顧客点・対応点では高い評価を得ることができています。その一方で、結果発表でもかなり突っ込まれていましたが、売上 2 位のチームが実は技術点では最下位だったりするのも、競技であるが故というところでしょうか。

優勝したチームは、マネージャはホワイトボードと付箋紙でタスク管理に注力して一切 PC を開かない、初動でマーケットプレイスからセキュリティエンジニア 2 名を競技時間まるまる買い占めて 1 チーム

12 名に増員[8]するなどという徹底したマネジメントとチームビルディングで勝利を掴みました。また、前日資料を元に、各サーバで起こりうる問題と対策の洗い出しなども行っていたようです。

脆弱なままのサービスは提供しないという選択

私たちのチームは「危険だとわかっているシステムは、まず安全性を確認してからオープンするのがお客様のためだ」という立場から、競技開始からまず初動作戦としてサービスを止め緊急メンテナンス画面を出して脆弱性の対策に振り切るという選択をしました。脆弱性を塞ぎ未然に防ぐことで繁盛レベルを稼ぎ、最終的な売上で他を引き離すという作戦です。負け惜しみでは無いですが、この選択そのものは間違っていなかったもののメリットを活かせませんでした。

まず事前準備の不足から脆弱性の対策に時間が掛かり、ずるずると「メンテナンス時間」が伸びてしまうという最悪なパターンに陥りました。まれに良く聞く話ですが、大変よろしくないです。これは「いつまでに・何ができなかったら・どうする」という復旧判断ができないから起こります。あらかじめ所持スキルにより担当システムの分担はしていましたが、こういった判断を最終決定する「責任者」を一人きちんと決めていなかったのが原因です。これもまれに良く聞く話ですね。恐ろしいことです。

マルウェアによる攻撃も絶対あるだろうという予想から、まずは Windows 端末にカスペルスキーを導入して守るという方針を立てたにもかかわらず、そもそもソフトウェアパッケージが大きくダウンロードに時間が掛かったり、Windows 10 に入れようとして対応 OS の都合で入らなかったりなど問題が多く発生しました。Windows 端末が使えなくなるとメール送受信ができなくなると言う焦りの空回りから結局カスペルスキー一つに 16 時ぐらいまで引っ張っています。

さらに、最初に「正しい動作」を確認せずにいきなり Wordpress のバージョンアップに特攻した結果、EC サイトの本体である Wordpress のショッピングカートプラグインの不具合を踏み抜き、正しい在庫数が表示されないという問題が起きました。Softening Day の答え合わせまで原因が全くわからず、kuromame6 による情報改ざん攻撃だと思い込み調査をしていました。在庫数がわからないため、適切なマーケットプレイスからの「仕入れ」もできず、結果として最後に一気に売上が落ちるというダメージを食らいました。無駄な調査に人員も取られました。

結果として、端末のマルウェアや業務アプリといったサービス停止で防げなかったインシデントの対応に追われたあげく、競技開始後から 3 時間もサービスを止めたにもかかわらず脆弱性への対応が不完全なままのサービス再開を余儀なくされました。そして、一時的に他のチームより繁盛レベルを高く維持できた時間帯がありつつも、売っているのに仕入れていないという問題でまったく売上を得ることができませんでした。

---

[8] その後「ガチャ」景品で 10 分間だけもう一人追加され、一瞬だけ 13 人体制でした。

次回勝つために

負けたら悔しいのが人間です。ならば次にどうするかを具体的に考えます。

kuromame6 からの講評では、技術点が最も高かったチームですら評価ポイントの 1/3 もクリアできていないようです。評価で重きを置かれるポイントは毎回変わるとはいえ、純粋に堅牢化の技術的そのものにもまだまだ根本的にひっくり返す余地がありそうです。

優勝チームを見習ったチームビルディング・マネジメントと、万全の事前準備による瞬発力のある堅牢化技術が合わさることで最強になれそうです。

何も考えずに脆弱性診断

全ての意志決定には十分な情報が必要となります。脆弱性診断に長けたメンバーが居るチームが今回の技術点 1 位を取っているとおり、やはり適切な脆弱性診断による現状把握が大事です。競技環境の構成がある程度わかっている以上、ツールでできるような脆弱性診断はなかば機械的に競技開始時点で開始すべきでしょう。

Softening Day での発表によれば、今回は EC サイトやコーポレートサイトが Wordpress ベースだったため専用の脆弱性スキャンツールである WPScan が有用だったようです。またサポート端末については一般的な脆弱性スキャナである Nessus だけでも色々発見できていたそうで、自分のチームに脆弱性診断の専門家が居なかった場合でもツールの使い方を予習しておくだけでかなりの成果が見込めそうです。

今回もそうだったように、Hardening Project ではその趣旨から「最近注目されている脆弱性」を取り上げる方向がありますので、それを発見できるような脆弱性診断方法を逆算して自動化して持って行ければ、さらに一歩先へ行けそうです。

あらかじめ仕込まれたマルウェアへの対応

今回のように、競技開始時点で脆弱性だけで無くマルウェア本体までが仕掛けられていた場合、本質的には「何も信じられない」という状態から始めることになります。たとえば既設のバックドアでリアルタイムに介入されることまで想定すると何もできなくなってしまいます。ですが、物理的に新しいサーバを追加できない競技環境では、競技を成立させるために「現実的なゆるさ」があり上手くそこに乗っかっていく必要があります。

何も信頼できない環境に安全な橋頭堡を築いていくのはある種のパズル的な要素がありますが、例えばシェルや SSH を信頼できる静的リンクバイナリとして持ち込んだり、脆弱性診断でなんとかしたり

といった手が考えられます。Windows であれば、ひとまず機械的にオープンソースのアンチウイルスソフトを導入すると言った手もありそうです。

「なにかされても被害を出さない」という封じ込めも有効そうです。全ての通信が通るルータは VyOS で手が出せるので、マルウェア自身への対策と合わせて実施するのが良さそうです。

作業環境の改善

競技環境はとにかくサーバの種類が多いため、ただ単純に SSH やリモートデスクトップで一台ずつログインして手作業でコマンドを売って回るといくらあっても時間が足りません。1 台 2 分の作業でも 13 台の CentOS サーバに一台ずつ廻るだけで 30 分近くも掛かってしまいます。

せっかく SSH が使えるのであれば、漏れたら最後また変えるしかないパスワードでは無く、参加者 PC から出さなければ絶対に漏れない公開鍵認証を利用することで、パスワードのコピペという手間からも解放され大幅に作業効率が向上します。例えば競技用の SSH プライベート鍵を共有しておけばよいでしょう。

こういった作業環境の改善は、かなり属人性が強く個人のノウハウが詰まっていることが多いのですが、そういったテクニックをあらかじめチーム内で共有することで作業の初速が上がります。初速を上げることは、最初の攻撃シナリオの実施までの時間が稼げることにもなり、何倍にも効いてきます。

Wordpress の管理画面などはブラウザからのアクセスが必要ですが、CentOS で VNC サーバを利用するなど Windows 端末に頼りすぎない作業環境を作り込むことも効果が大きそうです。

自らのサービスの正しい把握

サービスを正しく運用するためには、それを実現するシステムの正しい把握が必要不可欠です。今回の EC サイトであれば、Wordpress のどういったプラグインでどのように実現されているか、売り上げ管理システムとの関係はどのようなものか、売上と仕入れの関係はどうなっているのか、そういったことを専門で面倒を見る「サービス担当者」が必要です。

事前の作り込み

こういったことを考えながら、事前に公開されている情報をもとに、競技環境を予測してそれに対する事前準備を作り込むことが勝利への第一歩です。

1. 更新ファイルの用意
   Wordpress であれば最新版の ZIP パッケージや、競技環境のディストリビューションのアップデート RPM パッケージを持ち込むことで、素早いソフトウェア更新ができます。

2. ファイアウォールのルール
   通すべき通信を予測して iptables や VyOS の ACL 案を作り込んでおき、素早く不必要な通信を記録・遮断することで、マルウェアの被害を防ぐことができます。
3. 認証情報の共有
   パスワードの再設定、ユーザの整理は行うとして、変更後のパスワード一覧をあらかじめ乱数で作っておき、競技に使う SSH プライベート鍵をチームメンバーで共有しておくことで、これらの再設定を全て前もって自動化することができます。
4. 脆弱性診断の準備
   競技環境に即した脆弱性診断ツールを用意しておきます。細かいことですが、存在しているシステムの全体がわかっているのであれば、例えば診断先 URL などもあらかじめ設定しておくと貴重な時間を有効活用できます。
5. 開幕ぶっぱなしスクリプト
   これら用意した道具を全てのサーバに流し込む作業をスクリプト化しておきます。SSH が使える環境であれば Fabric が有用ではないかと感じています。

感想

長々と今回の h10v という Hardening Project の紹介をしましたが、いかがだったでしょうか。楽しさと悔しさが伝われば幸いです。

次回は 2016 年 6 月に沖縄で開催されそうな雰囲気です。

是非一緒に参加して優勝を勝ち取りましょう！

Hardening 2017 Fes レポート

TechReport 2017.12

「衛る技術」を団体戦形式で競い合うセキュリティ・イベント「Hardening Project」の 11 回目、「Hardening 2017 Fes」に参加してきました。

通常の Hardening は、実際の競技を行う「Hardening Day」と、振り返りプレゼンと攻撃チームからの解説、最終結果の表彰などを行う「Softening Day」の二日間で行われます。今回は「Fes」と銘打っている事もあり、全三日間の日程となり二日目に「Firming Day」が設けられました。募集段階では詳細は明かされず、「アンカンファレンス、ワークショップ、ラウンドテーブルなどの形」と公表されています。

会場もここ最近ずっと沖縄でしたが、今回は淡路島の淡路夢舞台国際会議場でした。いわゆる日本標準時（JST）の子午線が通る地域でもあります。

というわけで、我々「Team9 ￥>pin9」の闘いを、時系列で追っていきます。

Day-0 競技前日

Hardening では、前日夜にはチームメンバーが集まって競技当日に向けた最終方針決定などを行うのが基本です。そもそも沖縄の場合には当日朝着が現実的に不可能であることから、ほぼ全員が参加します。

実はまずこの段階でつまずき、某 IPA の某研修プログラムの日程の都合によって 7 人中 4 人[9]が前日打ち合わせに参加できませんでした。正直ここでポプテピピックコラを貼りたい気持ちで一杯になりましたが、来られないものは仕方無いので来られる人だけで打ち合わせです。

前日打ち合わせのやり方もチーム毎に大きく異なりますが、私達は KDL さんに縁のある店で複数のチームが集まってお酒もそこそこに集まりました。途中でマーケットプレイスの方々も遊びに来たりしています。

より本気度の高いチームは、Airbnb で広めの家を借りてチームメンバー全員がそこに宿泊したりもしていたようです。正直その発想が無かったので、各々で高速バスの都合が良い三ノ宮などに宿を取りました。とにかく朝が早い（高速バスの場合は 7 時にはホテルを出ていないといけない）ので、高速バス乗り場から近いところが良いです。

[9] チームの人数は、今回は 6 人もしくは 7 人です。そして全 16 チーム 108 人。

ちなみに、接続用にポート数の多いラックマウントスイッチを持って行ったのですが、空港で見事に預け入れ荷物の重量制限に引っかかり、まさかの手持ち搭乗になりました。

Day-1 Hardening Day

競技当日です。

チーム毎にあつまり参加費(メシ代)を支払って会場に入ります。会場にはネットワークケーブルが一本出てきている状態で用意されているので、持ち込んだスイッチ等で参加者の PC を接続していきます。私達のチームではありませんが、最近のノート PC はネットワークインターフェースを持っていないものが多く、持ち込んだ USB-NIC がうまく接続できないというチームもあったようです[10]。当初は 24 ポートのスイッチを使っていたのですが、今回のテーブル配置だと 8 ポートくらいの小さなスイッチを 2 箇所において近い方に接続する方が場所を取らず良かったようです。

まず競技内容の説明があり、それと並行してひっそりと 9 時間のカウントダウンもしれっと始まります。Hardening の場合は、このような競技説明にもヒントがあったりするため、接続確認を進めながらもきちんと耳を傾ける必要があります。

初動

さて、Hardening で重要とされる「初動」として、以下を迅速に進めます。

- 各サーバのログイン確認
- パスワード変更(一般ユーザ・管理者)
- 一般ユーザにログイン用 SSH 公開鍵を設定
- 各サーバの基本的なバックアップ
- マーケットプレイスの手配

パスワード変更や SSH 公開鍵の設定は、ある程度スクリプト化していたのですが、1 台混じっていた Ubuntu で上手く走らなかったりと若干の焦りもありつつも、ひとまずログインできるサーバについては実施しました。

バックアップについては、侵入を見越して対象と別のサーバ(踏み台 PC)への複製をしましたが、実は複製対象の確認が甘く、設定ファイルだけで Web コンテンツなどの複製ができていない状況でした。

[10] それを見込んで、うちは念のため複数ベンダーの USB-NIC を持ち込んでいました。

まさに「やるべきことをやりきることが重要」という Hardening の鉄則通り、バックアップ不備により、後のランサムウェアで痛い目に遭います。

その一方で、一般的な OpenSSH 環境でそのまま使える ProxyCommand 設定済み sshd_config をチーム内で共有して作業の迅速化を図ったりもしています。とにかくパスワードを打たないことが後々のキーロガー対策にもなるという予想から、基本的に全てこの段階から公開鍵認証でログインしてもらっています。後で他のチームのプレゼンを見て、このあたりのサーバ管理基本スキルについて事前訓練をやっておけば良かったなと感じます。

競技環境への誤解

これは思い返せばなんでこんな勘違いをしたのか判らないのですが、競技環境に接続した PC からはインターネットに出られないと思い込んでいました。そのため、事前に準備していた Slack や Google spreadsheet などを活用できず、土壇場で用意した付箋紙ベースでのプロジェクト管理を強いられることになり、無駄に苦労することになりました。

いやこれほんとなんで勘違いしたのかいまだに謎です。

競技環境上のサーバからはインターネットに出られませんが、接続している競技参加者の PC 自体はインターネットに出られたようです。

前任者による不正侵入

Hardening で重要なことは「想像力」です。与えられた情報全てから、どんなストーリーで何が行われているのかを推測していく能力が求められます。それが一つ現れるのが退職者による不正侵入というインシデントです。

各サーバのプロセス状況の確認から、「ametani」というユーザがログインして色々悪さをしていることが判明しました。当然ながら ametani なんてユーザは知らないので、ひとまずアカウントへのログインをパスワード変更で停止しつつ、「親会社の社長」や「総務部門」に問い合わせます。

その結果、現役のリモート作業者ではなく退職者であることがわかり、そのままアカウント停止処分と相成りました。

外部業者によるコンテンツアップロード

この上手く対応できた「ametani」氏インシデントと対称的なものが、外部業者からのコンテンツアップロードです。

あらかじめカバーストーリーとして、「外部業者にイケてるデザインを発注しているのでそれが競技期間中に届く」という事が明示されていました。その届いたデザインをウェブサイトに組み込むことで売上＝スコアが上がるというわけです。そして競技環境にあるファイルサーバにあるテキストファイルをよく読むと、ウェブサーバの一般ユーザ user6 でホームディレクトリにアップロードされる旨が書いてあります。

ここでおさらいですが、初動段階で一般ユーザのパスワードを一括で変更してしまっており、当然ながら外部業者がそのユーザでログインすることができなくなっています。「確認不足のまま現状を変更すると何かが起こる」というのは Hardening における王道パターンの一つで、見事にそれをぶち抜いた形です。

正しい対処としては、「別のリスクを許容しつつ、一般ユーザのパスワード自体は変更しない（そのかわり sudo 等を抑止）」もしくは「パスワードを変更して外部業者にその旨を連絡」の 2 パターンでしょうか。可能ならば後者が望ましいのは言うまでもないです。

ametani 氏の時は所属確認など注意深くできていたのに、かなり残念ポイント高いです。

ウェブサーバのランサムウェア

見事にやられました。

本来ならばバックアップから書き戻せば良いはずが、コンテンツディレクトリのバックアップミスにより、痛恨の支払いが発生します。まさにこれは現実世界でも同じ事が言えます。取っているはずのバックアップ、ちゃんと必要なものを全て取れていますか？

またランサムウェア復旧サービス（いわゆる支払い）に泣き付くときにもうっかりミスを重ね、10 万円の復旧サービス（今回は無関係）と、100 万円の復旧サービスの二つが用意されているのですが、100 万円の復旧サービスと明示されているにもかかわらず無駄に 10 万円の方を発注してしまいます。

会場には攻撃側チーム Kuromame6 からのヒントを代弁する Google Home[11]が居ましたが、みごと Google Home から「10 万円なんかじゃ復旧できないよ」と煽られることになります。

復旧サービスに申し込むと、勝手に不正侵入して暗号化されたファイルを復号してくれるのも悔しい限りです。

[11] これまでは Pepper でした。

Windows マルウェア

定番として Windows 端末がマルウェアにやられます。というわけでエンドポイント製品、今回は Intel 製品を導入しました。見事に検知していただき、エンドポイント製品の重要性と威力を思い知りました。

複数サーバで連携してその経路などを追いかけてくれるのは流石ですね、かなり良いです。

HTTPS 化

昨今の状況を顧み、カバーストーリーでも触れられていることから HTTPS 化が必須なので絶対やろうということに前日打ち合わせで決まっていたのですが、優先度判断が甘く最後の最後に回ってしまいました。おそらくこれによる売上低減が、中盤以降でスコアがまったく振るわなかった主要因の一つとみています。

HTTPS 化も、全て自前でやるという手法だけでなく、NEC InfoCage のような外部 Proxy 型の WAF であればそこに証明書を入れるだけで対応できたようです。正直マーケットプレイスに出ている製品の理解がまったく不足していたため、無駄に優先度を下げてしまったわけです。マーケットプレイスで購入を検討している製品は、どのような導入形態が取れるのか事前調査が勝利の鍵と思います。

壊れた Wordpress

コーポレートサイトでのリンク先が間違っているためアクセスが来ないという仕込みがあるのですが、この Wordpress の管理画面が壊れているため、まともにコンテンツが編集できないという問題がありました。

本当に Wordpress 自体が壊れているわけではなく、競技環境は本物のインターネットから隔離されているのですが、本物のインターネット上の Google CDN から取ってくる JS に依存するなど、このあたりを早めの時間帯で切り分けて対策できていれば良かったと思います。また、いっそ API での編集など別の手段を確保するという手もありかも知れません。

情報漏えい事故

Web コンテンツのディレクトリに、DB のバックアップなどがぽろっと置かれているという情報漏えい事故が仕込まれています。

Web コンテンツのチェック、初動での確認事項に入れたい項目の一つです。

また、今回は各チームが「子会社」という体になっていますが、その時点でスコアの高い一部チームに対して情報漏えいインシデントに対する「親会社からのお呼び出し」が掛かります。別の個室で行われる状況がメイン会場にも映し出され、まだ呼ばれていないチームは戦々恐々としていました。

ルータへのログイン手段

競技ネットワークで 2 台用意されているルータ（これも競技対象の中）へは、踏み台として用意された PC から SSH でログインするように設定されていますが、ここもまたシナリオへの想像力が求められました。

基本的に user1 という一般ユーザを利用して作業をしていますが、ルータ 1 へは素直にファイルサーバ上の用意されている PuTTY とその秘密鍵を利用して user1 にログインができます。判りづらかったのがルータ 2 へのログインで、やはりファイルサーバ上に用意された別の秘密鍵を利用しますが、ログインできるのは user2 だったりします。こういう罠がたくさん用意された競技環境、とても楽しいです（白目）。

競技終盤

まあ終盤になってくるとだいぶ焦りが溜まってくるわけですが、細かい対応不足が重なってスコアが伸びず、よりさらなる焦りを呼びます。

仕掛かり作業をやりきろうと言うことで、HTTPS 化などに今更のように手を付けたのもこのあたりだったりします。当然ながら、そんなスピード感だと現実では Google chrome に怒られてしまうわけですね。

競技終了

9 時間の競技を終え、レセプションホールでの立食パーティタイムです。

ここで運営側から競技時間の追加が告知され、「明日はまたゼロから」「引き継ぎをいかに詳細にするか」というキーワードが出ます。他チームとのスワップ辺りかなあと予想しつつ初日終了です。

Day-2 Firming Day

問題の 2 日目「Firming Day」です。

競技時間の追加は、「人事異動」として「リーダー以外が隣のチームに異動」することが発表されます。つまり、リーダー一人だけを残して別のチームの人が自分たちの環境を見ることになります。まさに引き継ぎ力が重要という通りです。

延長競技時間は 2 時間、私達はリーダーを残して隣のチーム「Team10 天一」に移って引き継ぎを受けました。どうも引き継ぎ資料が不十分で、パスワードが分からないなんてチームもあったようです。隣のチームは、競技環境に splunk を導入してログ分析をやっていたりと、色々各チームの特色が見えて価値の大きい 2 時間でした。

ちなみに、この日のランチがカレーでしたが、本当に美味しかったです。

Day-3 Softening Day

Softening Day では、まず各チームがそれぞれの取り組みをプレゼンします。

特徴的だと思った取り組みをいくつか紹介します。

原価率などの戦略

仕入れ原価などを差し引いた売上額がスコアと言うことで、競技環境の EC サイトに並んでいる商品の原価率などに売上戦略に注目したチームが複数有りました。特に高額商品の「アナゴ」をたくさん売ったチームはスコアが高かったようです。なかには、新しい商品としてネギを新規に売り出したチームもありましたが、惜しくも「お客さん」には刺さらなかった模様です。

その一方で、「32 億円の誤発注」という伝説的なやらかしがあったチームもありましたが、それをきっかけにチームの雰囲気が良くなり笑いの絶えない職場になったりと、なにが幸いするか判りません。

社歌

メール担当者が、出すメールのフッタに社歌を入れていたチームがありました。

この社歌は、翌月に東京で行われた振り返り会にて無事披露されました。

ツール

各チームいろいろ試行錯誤が見えましたが、タスク管理は Trello が良さそうという感じです。

Backlog や GitHub issues というチームもありました。

事前訓練

先ほど少し触れましたが、競技環境を模したサーバ群を用意して、基本的な作業や初動対応などを訓練したチームが複数有りました。

確かに Hardening の競技環境は「踏み台」の存在など癖のある部分も少なくなく、これは必ずやっておいた方が良さそうです。

Kuromamoe6 からの解説と表彰

各チームからのプレゼンの後、攻撃チーム Kuromamoe6 からの答え合わせの時間です。

そして、最終スコアの発表と表彰が行われます。私達 Team9 は辛うじて最下位ではない 15 位でしたが、得るものは多かったので実質優勝です。

動画公開

そして、ここまで全て YouTube にて動画が公開されています。是非みましょう。

⇒ https://www.youtube.com/watch?v=me4rM9OVtNg

Day-4

Day-3 を夜までゆっくり楽しみたかったので後泊しています。

この日は、Hardening の疲れを癒すべく有馬温泉に寄って帰りました。

今後に向けて

最近すっかりサーバーレス屋と化して、サーバの運用をあまりやっていない状況だったのですが、改めてサービス運用・サーバ運用の勘所を復習する良い機会になりました。

今後また Hardening に向けてやっておいた方がいいなーと思ったことをちらほら並べます。

- 事前訓練で競技環境の癖に慣れる
- バックアップなど、現実だったらやっていることをきっちり意志決定
- 前日打ち合わせは全員参加、場合によっては宿泊も Airbnb で一緒にする
- きっちりタスク管理、優先度判断
- 素人なりにでも良いから最低限の脆弱性診断
- splunk 等によるログの一括管理
- コンテンツの全チェック
- ビジネスの利益率向上施策についての議論

● なんだかんだで世の中 Wordpress なので WP 力を付ける

これらは Hardening だけでなく実世界でのサービス運用に通じるのは言うまでも無いです。

最後にあらためて、Hardening Project という企画を動かしている全ての人達に感謝します。


この炭酸泉は飲むことができます。
右のレバーを押してお飲みください。
炭酸泉源

私もセキュリティ・キャンプに参加したい！

TechReport 2017.08

みなさんは「セキュリティ・キャンプ」をご存知ですか？

セキュリティ・キャンプは、情報セキュリティおよびプログラミングに関する高度な教育を若者に対して実施することで、いわゆる「高度 IT 人材」の発掘と育成を目指そうという事業です。2004 年からスタートし、それから名前[12]や主催組織が紆余曲折ありましたが、2012 年からは現在の民間企業などで構成されるセキュリティ・キャンプ実施協議会（2018 年より一般社団法人セキュリティ・キャンプ協議会に組織変更）と、経済産業省系の独立行政法人情報処理推進機構（みんなだいすき IPA）の共同主催という形をとっています。

セキュリティ・キャンプといってもアウトドアでキャンプをするわけではありません。中心となる全国大会を始めとした様々なイベントが行われ、それらを総称してセキュリティ・キャンプと呼んでいます。

この記事では、セキュリティ・キャンプについて紹介します[13]。

セキュリティ・キャンプ 全国大会

毎年 8 月に東京近辺で行われている 4 泊 5 日の合宿イベントで、一般に「セキュリティ・キャンプ」と言った場合にはこの全国大会を指します。だいたい夏コミの直前か直後の週で金曜～土曜が被ることもあります。今年（2017 年）は幸い夏コミ明けの月曜日からのようです。

この後に紹介するように内容の濃さもさることながら、会場までの旅費や当日の宿泊費など受講に関する費用をまるっと出してもらえるということもあり、全国津々浦々から参加者が毎年集うのが全国大会の特徴です。また、5 日間のうち中 3　日間は朝から晩まで講義ということで、こちらも全国津々浦々からセキュリティあるいはコンピュータサイエンスにおける様々な分野の専門家を 50 人以上も集めています。総勢 100 人以上の専門家と専門家の卵が全力でぶつかる場として、ドラゴンボールの「精神と時の部屋」のように言われることもあります[14]。

---

[12] 2009 年～2011 年は「セキュリティ&プログラミングキャンプ」

[13] ステマ回避のため最初に書いておくと、2017 年時点で筆者もセキュリティ・キャンプの講師およびプロデューサー（旧トラックリーダー）を担当しています。

[14] そろそろ通じなくなっているという噂も聞きますが……

参加者は 4 月から 5 月末まで募集され、そこから応募内容に基づいて選考されます。ここ数年は 50 人程度でしたが、2017 年から 80 人程度に増員されました。それに伴い、会場も幕張から府中に移動になっています。選考通過者全体では平均年齢が上がっている一方で、2017 年には初めて小学生が選考を通過しました（しかも 2 名！！）。

2017 年から 50 人前後から 80 人前後に大幅に人数が増えたことに伴い、「選択コース」と「集中コース」に大きく分けて実施されることになりました。

選択コース

この数年「トラック制」として実施されていた全国大会の形式をそのまま引き継ぐのが、この選択コースです。

様々な分野の講義（3～4 時間）が 5 つのトラックで並行して用意され、参加者はそこからどの講義を受講するか希望することができます（部屋や講義体制の都合で必ずしも希望通りにならないこともあります……すみません……）。2 枠もしくは 3 枠を連続で確保し、食事を挟んで最大 11 時間ぶっ続けという「選択ってなんだっけ」という講義もあります。

自分が既に得意な分野の講義ばかりを選んでも良いし、あまりよく知らなかった分野に手を出してみるきっかけとしてザッピングするのも良いです。参考までに、2017 年の二日目の時間割を紹介します[15]。様々な分野の講義があるなと感じていただければと思います。

[15] https://www.ipa.go.jp/jinzai/camp/2017/zenkoku2017_jikanwari.html

時間	内容	トラックA	トラックB	トラックC	トラックD	トラックE
08:30-12:30	専門（1）	**A1** PowerShellベースのマルウェアとその防御手法 講師： 凌 翔太	**B1** DOS攻撃用FPGAを作ろう 講師： 内藤 竜治	**C1** ブラウザの脆弱性とそのインパクト 講師： Masato Kinugawa 西村 宗晃	**D1** Linuxカーネルを理解して学ぶ脆弱性入門 講師： 小崎 資広	**E1~3** BareMetalで遊びつくそう Raspberry Pi 講師： 西永 俊文
12:30-13:20	昼食					
13:30-17:30	専門（2）	**A2~3** AIのデータ汚染を検知しよう 講師： 松田 健 佐藤 公信	**B2** 組込みLinuxクロス開発スタートアップ 講師： 海老原 祐太郎	**C2** ID連携と認証基礎 講師： 林 達也 真武 信和	**D2~3** カーネルエクスプロイトによるシステム権限奪取 講師： 木村 廉	**E1~3** BareMetalで遊びつくそう Raspberry Pi 講師： 西永 俊文
17:30-18:50	夕食					
19:00-22:00	専門（3）	**A2~3** AIのデータ汚染を検知しよう 講師： 松田 健 佐藤 公信	**B3** PF_PACKETで仮想IP環境を自作してパケットの理解を深めよう 講師： 小俣 光之	**C3** Vulsを用いた脆弱性ハンドリングとハッカソン 講師： 神戸 康多	**D2~3** カーネルエクスプロイトによるシステム権限奪取 講師： 木村 廉	**E1~3** BareMetalで遊びつくそう Raspberry Pi 講師： 西永 俊文

朝の 8 時半（実際にはその前に朝食）には講義開始して夜の 22 時まで詰め込まれているというあたりも「合宿」ならではというところですね（白目）。

これらの自分で選んだ専門講義のほか、全員が共通で参加する全体講義として「セキュリティ基礎」「特別講演」「BoF」「企業プレゼンテーション」などがあります。1 日目・5 日目は全体講義のみで、中三日は集中講義がメインです。

講義の他にも少人数のグループに分けられて課題のテーマについて 5 日間で議論する「グループワーク」もまた全国大会の重要なイベントです。講義毎に参加者が入れ替わる選択コースで、ずっと同じグループで活動ができる貴重な機会なので有効活用して下さい。ただしグループワークのために用意されている時間は限られ、いかに健康で文化的な生活を保ったままグループワークの準備を進めるかも試されます。

それぞれの講義には、講師のほか 2 名前後の「チューター」が付きます。彼らはセキュリティ・キャンプの卒業生であり、講師よりも受講者に近い立場として、講義だけでなく 5 日間の受講者の生活を朝から晩まで様々な面から支援してくれます。

集中コース

2017 年から新しく新設された集中コースでは、3 つのテーマに 10 人程度の参加者が 3 日間集中して取り組みます。初回は「言語や OS を自作しよう」「セキュアな CPU を自作しよう」「Linux 向けマルウェア対策ソフトウェアを作ろう」という 3 つのテーマが用意されています。

共通講義やグループワークなどは選択コースと同一とはいえ、一つのテーマを突き詰めていく、いわばもう一つの全国大会が何を成し遂げるのか、今から楽しみです。

セキュリティ・キャンプ 地方大会

選考倍率が高い全国大会だけでなく様々な機会を提供するため、「セキュリティ・ミニキャンプ in ○○」という地方大会を各地域で実施しています [16]。

誰でも参加が可能な「一般講座」と、25 歳以下の学生のみ参加できる「専門講座」の二日間開催という形態が多く、一般講座の存在が全国大会と最も異なるところです。また、地元の参加者を想定していることもあり、交通費や宿泊費については自己負担であったり一部負担であったりすることもあります。

2016 年度には 9 回実施され、2017 年度にも 10 回予定されています。まだまだ C92 時点で募集中の宮崎・山梨を含めて 8 回応募する機会がありますので、興味を持った方は是非応募してみて下さい。比較的応募者が地域に閉じていることもあり、全国大会と比較して応募倍率などかなり敷居が低いです。2016 年度の地方大会の様子は、@IT にて連続連載 [17]が組まれていたので、そちらを読むとどんな雰囲気か判ると思います。

---

[16] 「セキュリティ・キャンプ九州実施協議会」がある福岡のみ「セキュリティ・キャンプ in 福岡」

[17] http://www.atmarkit.co.jp/ait/series/3922/

その他のセキュリティ・キャンプ活動

セキュリティ・コアキャンプ

2017 年全国大会と合わせて、全国大会の OB/OG 向けの成長機会として新しく実施することになった枠組みです。かなりキャッチーなタイトルが並んでいるので紹介しておきます [18]。

> 『中津留勇 マルウェア解析 LIVE 2017 ～CAN'T STOP ANALYSIS～』
> 『日帰りバグハンター合宿』
> 『キャンプ講師になるためのコンテンツハッカソン』

ちょっと尖りすぎて、応募者ドン引きで出足が悪かったという笑い話もあったりなかったりしました。

セキュリティ・ジュニアキャンプ

全国大会などは 22 歳以下の学生が対象ですが、さらに若い年代向けのイベントとして中学生以下を対象とする「ジュニアキャンプ」を高知で毎年実施しています。

最近はロボットカーをテーマに、セキュリティを絡めて二日間で RaspberryPi ベースのロボットカーを実際にコースで走らせるところまで手を動かしてもらうという内容になっています。後半は、互いの参加者のロボットカーに「いたずら」をするという擬似的な攻守合戦のような形式で、コンピュータにおけるセキュリティの重要さ、どのような攻撃がありそれからどうやって守るかを、身をもって体験しています。

セキュリティ・キャンプアワード

セキュリティ・キャンプ全国大会修了生のその後の活動を発表してもらい、修了生同士の活躍を互いに強化していくための枠組みとして、年一回の「キャンプアワード」という場を用意しています。

毎年 50 人前後の「キャンパー」を排出する全国大会ですが、その中でも先進的な活動を継続して続けている人のうち、一次審査を通過した 5 人がその場で最終プレゼンを行い、最終的な賞を授与されます。2017 年 3 月に行われたキャンプアワードの様子が@IT で記事 [19]になっています。

正直めっちゃレベル高いです。

---

[18] http://www.security-camp.org/event/corecamp2017.html

[19] http://www.atmarkit.co.jp/ait/articles/1704/11/news011.html

GCC：Global Cybersecurity Camp

2018 年よりコアキャンプを発展する形で実施されるのが、この Global Cybersecurity Camp――通称 GCC です。「国籍・人種を超えた専門知識のあるグローバル人材の育成」と「国境を超えた友情とゆるやかなコミュニティの形成」を目的として開催されます。日本のほか韓国、台湾、シンガポールにおける若手エンジニアを対象に、4 カ国の講師がお互いにセキュリティ技術に関する講義を行う、まさにグローバルなセキュリティ・キャンプです。

応募対象は 25 歳以下のセキュリティ・キャンプ全国大会の修了生[20]で、日本から 7～8 名が選考されます。参加費用や交通費・宿泊費などは全て無料です。

セキュリティ・キャンプに参加しよう

セキュリティ・キャンプに参加するにはいくつかの方法があります。

正攻法で全国大会に応募する

あなたが 2019 年 3 月 31 日時点で 22 歳以下の学生であれば、来年度の全国大会に応募するのが一番の近道です。参加費用を全て負担してもらうことができ、手厚い講師陣による熱い講義をうけることができます。

その一方で、全国大会は応募倍率も高く、その応募課題のレベルも高いのですが、きちんと努力をしてそれを表現できれば通過できるのも事実です。講師陣・修了生からのメッセージ[21]という Togetter にまとまっているので、是非参考にしてください。

上の Togetter にも載っていますが「応募課題への回答は選考担当者とのコミュニケーション」に尽きるので、正誤を恐れず、自分が何を考えてどう手を動かしているのかを書いてください。「解答」だけを端的に書かれても、選考担当者は超能力者ではないので、どんなことを考えている人なのか判らないというのが正直なところなのです。そういう「もったいない」回答が少しでも減ればと思います。

---

[20] ただしジュニア限定ネットワークゼミの修了生を除く

[21] セキュリティ・キャンプ全国大会へ応募する上での講師陣・修了生からのメッセージ https://togetter.com/li/1104730

全国大会のジュニア限定ゼミに応募する

2018 年より小中学生限定のジュニア限定ゼミ（2018 年は「ジュニア限定ネットワークゼミ」）が用意されました。

この枠はセキュリティ・キャンプ全国大会の中でも特別な扱いで、本来であれば全国大会の修了生は受講者としては応募できず、チューターや講師でしか再度の参加はできないのですが、このジュニア限定ゼミの修了生は、もう一度全国大会の一般受講者として応募できます。

ジュニア限定ゼミとそのほかのコースは併願も可能なので、もし自分が小中学生であれば、あるいはあなたのお子さんが小中学生であれば、チャレンジすることを強くおすすめします。

まずは地方大会に参加する

地方大会は、年齢制限が緩かったり選考倍率が低かったりと、全国大会と比べて参加しやすい傾向にあります。また、交通費をいとわなければ実施回数も多くあります。ちょうど本書が夏コミということもあり、次の全国大会の募集がある 4 月まで待ち遠しい方は地方大会への参加を考えてみてください。2017 年度の応募結果によると選考通過者の 31.7%が地方大会の参加経験者のようで、全国大会に向けたステップとして一定の成果が出ているようです。

また、一般講義については基本的に選考無しでそのまま入れるので、オトナな人は地元で地方大会が行われたときは是非お越しください。

大人のセキュリティ・キャンプ

基本的に、22 歳だとか 25 歳だとか以下の学生のみが参加できるセキュリティ・キャンプですが、実は大人が参加する方法があります。それが、セキュリティ・キャンプ協議会への参加です。協議会は、年会費 1 口 50 万円で参加組織を募集しており、会員になると以下のような特典があります。

1. 全国大会の見学と BoF 等への参加
2. インタビューや取材時におけるバックボードロゴ掲示
3. インターンシップの取り組み

そうです、普通であれば学生のみが受講できる全国大会の講義を、会員企業になると堂々と見学することができるのです。あくまで見学ということで学生同等の扱いではありませんが、5 日間の研修費用と考えれば、そこで得られる学生や講師との関係構築などを含めてかなり価値が高いのでは無いでしょうか。

協議会の会員企業が増えることで、地方大会を含めた様々な仕組みの自由度が増します。是非、大人の皆様も、会員企業になって全国大会に参加していってください [22]。

そして 2018 年には、待望されていた個人メンバー（賛助会員）が設定されました。年会費 10 万円で全国大会や地方大会の見学が可能になります。個人で気軽に出せる金額ではありませんが、純粋に教育コンテンツとして見れば破格と思います。こちらも是非おすすめです。

おまけ

一年を通してセキュリティをテーマにしたものづくりを行う通年ハッカソン「SecHack365[23]」もよろしくお願いします [24]。

---

[22] 繰り返しますが、筆者は関係者です。

[23] https://sechack365.nict.go.jp/

[24] こちらも筆者は(同上

クラウド四方山話「サーバーレスアーキテクチャ」

TechReport 2015.12

IT 業界で 2015 年もっとも流行ったバズワードが「IoT」だとすれば、パブリッククラウド業界で 2015 年後半もっともバズったのが「サーバーレスアーキテクチャ」ではないかと思います。特に 10 月に行われた re:Invent（AWS のカンファレンス）あたりから一気に叫ばれるようになったように感じます。

他のバズワードでもありがちな明確な定義が無いマーケティング用語ですが、2012 年あたりから「Serverless」という単語が明確なフレーズとして使われ始め [25]、AWS Lambda がそれをアピールに利用したことで一気に広がったようです。

「サーバ」でイメージされる範囲が広いのでサーバーレスアーキテクチャに対しても様々な理解があるようですが、大きく二つのパターンに整理できるようです。

1. アプリケーション実行環境として、サーバという抽象化単位を隠蔽したマネージドサービスを利用するパターン
2. アプリケーションのビジネスロジックをフロント（ブラウザやスマホ・デスクトップアプリ）に移したうえで、データストアなどのコンポーネントを直接利用するパターン

前者はソフトウェアのアーキテクチャ、後者はシステム全体のアーキテクチャという点で視点そのものが大きく異なります。きっかけとなった AWS Lambda は前者の認識をもとに「サーバーレス」を謳っていますが、クラウドコンピューティング全体の方向性としては後者における「サーバーレス」化も進んでいくのではないかと考えています。この記事ではそれぞれの定義における「サーバーレス」が現状どういったものであるかを整理し、課題と方向性を議論します。

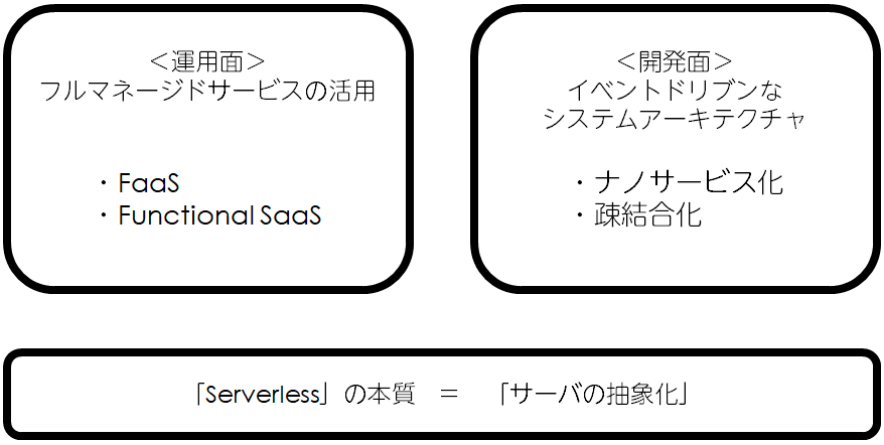
---

[25] おそらくこのあたりでしょうか。

Why The Future Of Software And Apps Is Serverless

https://readwrite.com/2012/10/15/why-the-future-of-software-and-apps-is-serverless/

「サーバーレス」については、既刊「Serverless を支える技術」にてより掘り下げて整理をして紹介しています。ここではその一節を引用します。



計算機の抽象化によって「サーバに依存しない」という制約が登場し、それによってスケーラビリティの確保が容易となり、それと相まって FaaS や Functional SaaS のようなクラウドサービスが登場しました。これらをイベントドリブンなシステムアーキテクチャによって最大限に活用すること、この全体が「サーバーレス」という技術ムーブメントの全体像です。

「サーバーレス」という言葉からイメージしやすいのはフルマネージドサービスによるサーバからの脱却という運用面ですが、実際に世の中を変えているのは開発面におけるイベントドリブンアーキテクチャによるソフトウェア開発自体のパラダイムシフトです。そしてその根底では「サーバ」というアプリケーション実行環境の抽象化が進んでいます。

サーバーレスアーキテクチャと PaaS

「サーバーレス」化を盛んに言い始めたのは AWS Lambda ですが、その流れの発端は 2008 年にリリースされた Google App Engine です。どちらも、下にあるサーバのプロビジョニング（仮想サーバインスタンの作成と実行環境の構成）を隠蔽してステートレスなアプリケーションを動かすことができるアプリケーション実行環境です。どちらも一見似ていますが、その発展の仕方は大きく異なります。

Google App Engine はもとよりウェブアプリが大前提となっています。時刻をトリガーとするスケジュールタスクはありますが、Cloud Pub/Sub からのプッシュや Cloud Storage の変更通知などの Google Cloud Platform 内でのコンポーネント間連携もすべて HTTP 経由で呼び出されます。

対して AWS Lambda は 2014 年に最初にリリースされたときには、Amazon S3 や Amazon DynamoDB の更新や Amazon Kinesis 経由でのメッセージをトリガーとして、非同期に実行したアプリケーションを実装するための、いわばコンポーネント間連携の枠組みでした。それが、まず 4 月に同期呼び出しが実装されるとともに AWS Mobile SDK を経由してモバイルアプリから直接呼出せるようになり、API のバックエンドとしての道が開けました。さらに、7 月に Amazon API Gateway の登場で普通のウェブ API が AWS Lambda で実装できるようになり一気にその可能性が注目されるようになりました。

最終的にはどちらも、ステートレスなアプリケーションを受け取り、上手い具合にプロビジョニングされた実行環境を提供してくれ、外部からの HTTP リクエストへの応答や他のコンポーネントと連携可能という、似たような立ち位置に置かれています。

サーバーレスアーキテクチャと PaaS の違いが良く議論されます。Heroku に代表されるいわゆるアプリケーション PaaS という分野は、技術的に十分成熟しプロビジョニングやオートスケールがマネージドサービスとして自動で行われることは当たり前となっています。ステートレスなアプリケーションが PaaS 上でスケールさせやすいということも以前から知られており、代表的な PaaS 事業者である Heroku の共同創業者 Adam Wiggins 氏が「The Twelve-Factor App」[26]として具体的な設計方針をまとめています。

つまり、実行環境がステートレスであることを強制しているかどうかにかかわらず、そもそもスケールさせやすいアプリケーションを書くにはステートレスであるべきといえます。このようにアプリケーション自体においてそもそもサーバという単位を考えることはなくなりつつあり、いわゆるサーバーレスアーキテクチャと従来からの PaaS に大きな違いは無いように見えます。

[26] http://12factor.net/ja/

アプリケーション実行環境としてのサーバーレス化

それでは何が「これがサーバーレスアーキテクチャだ」と呼べるような実行環境の特徴かといえば、それは実行環境のプロビジョニングとオートスケールが真に隠蔽された結果、実際に利用したリソースに限りなく近い課金が行われることではないかと考えます。

Google App Engine ではあくまでインスタンス単位で動作しますが、分単位で自動オートスケールと課金が行われるため、余剰リソースがほとんど無く実際に利用したリソースに近い課金が行われます。あくまでリソースの追加・削除がインスタンス単位でしか行えないだけだという見方もできます。AWS Lambda に至っては、実際にアプリケーションが消費した CPU 時間に対して 100 ミリ秒単位で課金されます（Google App Engine もプレビュー時代は CPU 時間に課金されていました）。

利用者視点で見ると、この消費した CPU 時間への課金というところが、サーバーレスアーキテクチャの一番の特徴と言えます。どんなに賢いオートスケールのアルゴリズムであっても、まともな応答時間でサービスを提供するためには余剰リソースが必要になるのに対して、消費した CPU 時間への課金であればいくら裏で余分に余計にリソースを確保していたとしても使わなければ良いので、もはや裏側でどんなオートスケールが走っているかを（正常に動いている限り）考える必要は一切無くなります。

その一方で、読込と初期化に何秒もかかるような大規模なフレームワークを使うにはまだまだ課題が大きそうです。Google App Engine ではダミーのリクエストを投げて「サーバを暖める」ための Warm up Requests が提供されています。一方でオートスケールを完全に隠蔽した AWS Lambda では、使用頻度の低いアプリケーションの初期化に時間が掛かってしまうという報告もあります [27]。これは、高レベルな AWS コンポーネントとの連携を前提とした、初期化処理がほとんど必要無いような Microservice での利用を前提としているが故の割り切りとも言えるでしょう。

[27] AWS Lambda for Java
https://medium.com/@quodlibet_be/aws-lambda-for-java-5d5e954d3bdf#.drsn6gtv9

ほんとうのサーバーレスアーキテクチャにむけて

「確保した量」から「使用した量」へのシフトによりサーバという単位を完全に忘れようとするのがアプリケーション実行環境におけるサーバーレスアーキテクチャですが、提供されたサービスの組み合わせだけでサービスを組み立てることで、「アプリケーションサーバ」という存在自体を無くしてしまう方向での技術革新もだいぶ進みました。

スマートフォンアプリや SPA（Single Page Application）の普及により、サーバは認証と REST API によるデータの送受信だけを提供すれば良いという時代になりつつあります。REST API はその性質からデータストアと相性が良く、さらに各クラウド事業者が提供する認証基盤の SDK と組み合わせることができます。例えば Facebook でログインして Amazon Cognito の ID 連携を経由して DynamoDB や S3 上の自分の領域に直接データを保存するといったことが実際に可能です [28]。

パブリッククラウドが提供するデータストアコンポーネントは全体として圧倒的なスケーラビリティを持っているため、アプリケーションレベルごときの突発的負荷などにも軽く対応できます。そのため、実際にはバックエンドでは AWS Lambda などでアプリケーションが動いていたとしても、ユーザとの接点はデータストアと直接やりとりするようなタイプのシステムが増えていくと予想しています。

API としてユーザに提供したいインターフェースと、実際のデータモデルの間を埋めるものとして、Amazon API Gateway や Microsoft Azure API Management のように API 管理層が間に挟まり、データ形式の変換、キャッシュなどを行うコンポーネントが提供されています。Amazon API Gateway はもっぱら AWS Lambda の呼び出しという味方をされることが多いですが、本来はもっと大きな可能性を持っています。

---

[28] S3 + Lambda + Cognito を使って、簡単お問い合わせシステム構築 - Qiita
http://qiita.com/takuma_yoshida/items/3bd3af8b09307d25981a
Cognito 認証による DynamoDB FGAC #アドカレ 2015 | Developers.IO
http://dev.classmethod.jp/cloud/aws/dynamodb-fgac/

サーバーレス時代の ID 連携

もう一つ重要なものが、ユーザの識別と認証です。

従来のウェブアプリなどでは、暗号化したブラウザクッキーやセッション DB などのサーバ側アプリケーションのみがアクセスできるセッション変数などを使うことで、リクエストをまたいでユーザを識別していました。そして、アプリケーション自体はデータストアに対する全ての操作権限を持ち、アプリケーションがユーザの ID をみて可能な操作のみを実装していました。ところがアプリケーション処理がユーザ側の端末で行われるようになると、アプリケーションの実装自体を信頼できなくなります。

古くから OpenID で ID 連携を提供し現在も OpenID Connect など自ら認証基盤と ID 連携を提供する Google をはじめとして、外部のソーシャルログインとも連携できる Amazon Cognito、Microsoft Azure Active Directory B2C など、各事業者は他のコンポーネントと ID 連携できるような枠組みを揃えています。

これは、アプリケーション自体に権限を持たせるのでは無く、「アプリケーションを利用しているユーザ」ごとに信頼して権限を持たせるという流れに沿ったものです。アプリケーションの上で ID とパスワードを直接入力するのでは無く、ブラウザ等を介して認証を提供するプロバイダーと直接 ID・パスワードのやりとりをして、あらかじめ必要最低限の権限だけを与えられたアクセスキーのみをアプリケーションに渡します。これでアプリケーションはどれだけ頑張っても、あらかじめ認可された範囲の操作しかできなくなります。

セキュリティの基本は「最小権限の原則」ですが、スマートフォンを筆頭にクライアント側アプリケーションの流れが来ている中、これからの数年で認証基盤と ID 連携を中心に「最小の権限だけを与えるとはそもそもどういう事か」という見直しが進んでいくものと思います。

おまけ：「薄く広い」アーキテクチャ

プロビジョニングの話が出ましたが、最近私が注目しているのが「薄く広い」アーキテクチャです。判る人には Google BigQuery と言えばピンと来るかも知れません。Google BigQuery を簡単に紹介すると、RDBMS っぽく SQL でクエリが書けるデータストアですが、膨大な件数のテーブルに複雑なクエリを投げても一瞬で結果が帰ってくると言う夢のような代物です。ただしその分制約も多いです。

Google BigQuery がやっているのは究極の力業です。データを数万台以上の膨大な数のサーバに分散保存し、クエリもその全てのサーバで一気に走らせてフルスキャンした結果を集約するというものです。これは Google 自身が持つ巨大なデータストアを共有で相乗りしているから可能となるスケールアウトの手法です。あまりぴったりくる訳語を見かけなかったので、ひとまず私はこのようなアーキテクチャを「薄く広い」アーキテクチャと呼んでいたりします。

サーバーレスアーキテクチャも消費 CPU という結果ベースの課金であることが特徴と書いてきましたが、BigQuery の課金モデルも同様に「実際にスキャンしたディスクの容量」と大変分かりやすくなっています。

一番重要な点は、BigQuery のように「薄く広い」アーキテクチャでしか実現できないことがあり、これらはパブリッククラウド事業者のような大規模クラスタを共有できる状況でしか実装ができないというところです。今のところはまだありませんが、アプリケーション動作環境でも「薄く広い」アーキテクチャでないと実現が難しい「何か」が出てきたときが、本当の変革の時なのでは無いかと思います。その点で、IoT 業界がやろうとしていることの規模感には期待をしていたりします。

サーバーレスで実現するストリーム処理徹底活用術

サーバーレス技術の代表的ユースケースの一つが、継続して発生する「ストリームデータ」を低遅延で処理するストリーム処理です。小さい処理を Pub/Sub でパズルのようにつないでいくと、様々なシステムを実現することができる、まさにサーバーレス向きのアーキテクチャです。

その一方で、代表的ユースケースの一つとは言え、もとより技術的困難さが多い分野ということで、様々な苦労を隠蔽してくれるはずのサーバーレス構成において依然としてつらい状況に直面することが多くあります。

というわけで、今回は AWS や Azure におけるサーバーレスストリーム処理を解説します。

ストリーム処理の用途

ストリーム処理は、いわゆる「リアルタイムで何か知りたい」という要求に対応するデータ処理です。具体的には以下のような用途で広く使われています。

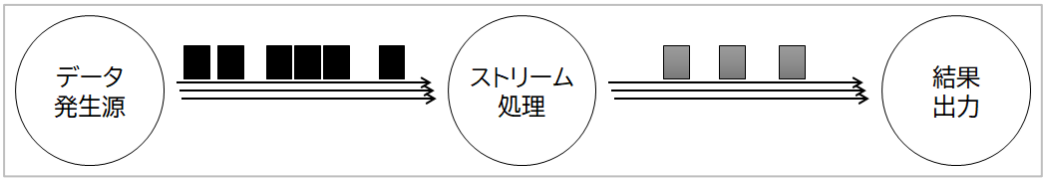
- アクセスログのリアルタイム解析
- 異常検知（例：不正ログインの監視・対策自動化）
- SNS（例：Twitter のタイムライン処理）
- IoT のセンサーデータの集計（例：異常動作を検知して交換手配、現在位置の推測）
- 株価変動に基づく自動売買
- サーバ監視のメトリクス集計
- CQRS（コマンド・クエリ責務分離）アーキテクチャにおけるコマンド（書込）処理

このような分析処理などを一日などまとまった単位で行うと、どうしても大量のデータを処理する必要があり時間も掛かります。ストリーム処理では後ほど紹介するデータの不完全さなどいくつかの制約に目を瞑ることにより、小さなデータ処理の繰り返しに分解することができます。これによってすぐに結果をえられ、それに基づいたアクションを素早く起こすことができます。

さらに、スケーラビリティの観点やサーバーレスの普及により CQRS アーキテクチャが注目されているため、データ分析用途以外においても今後ストリーム処理はより広く使われていくと予想しています。

ストリーム処理の基本

ストリーム処理をとてもざっくり言うと、無限に発生するデータに対して何らかの加工や集計をおこない、少ない遅延で何らかの結果を出力するというものです。



一般的なデータ処理と比較すると、いくつかの大きな違いがあります。

- 小さなデータが届くたびに何らかの処理を行い、それまでに届いたデータに基づいて、何らかの結果を出力します。
- データ発生源からデータが届くには時間が掛かる。
- データの発生順と入れ替わって異なる順番でストリーム処理に届く、あるいはデータの到着にとてもとても時間が掛かる場合がある。また、届いていないデータがあることを確認できない。
- 全てのデータが揃っていない不完全な状態でも、結果を出力しなくてはならない場合は、その場合は近似値となる。

そのため、イベントが発生した時刻（Event Time）と、処理を行った時刻（Processing Time）という 2 つの時刻 [29]に分けて、必要な整合性（もしくは割り切り）を確保する必要があります。また、遅延が前提となる処理であるため、いわゆる組み込み開発系で言われるようなリアルタイム処理との違いを尊重して、ニアリアルタイム処理と呼ばれることもあります。

ストリーム処理は、大きく 2 種類の道具で構成されます。一つは演算処理を実際に行うストリーム処理エンジン、もう一つはデータ発生源とストリーム処理エンジンを接続するメッセージングサービスです。さらにストリーム処理エンジンは、イベント一件ずつ逐次的に「真のストリーム処理」を行うイベント個別型と、時間や件数でイベントを集めてかたまり毎に処理を行うマイクロバッチ型に分けられます。今回扱うのは全てマイクロバッチ型です。

ストリーム処理は、これだけで厚い本が何冊も書けてしまう非常に面白い技術領域なので、是非追いかけてみてください [30]。

---

[29] 様々な遅延が想定される場合は、「システムへの到着時刻」も考慮する場合もあります。

[30] 入口としては、@kimutansk 氏の以下の記事やスライドが分かりやすいです。
https://qiita.com/kimutansk/items/60e48ec15e954fa95e1c

AWS におけるストリーム処理

AWS でストリーム処理を提供するサービスはいくつかあります。

メッセージングサービス	ストリーム処理エンジン
Kinesis Data Streams EMR（Apache Kafka） ※ Kinesis Data Firehose	Kinesis Data Analytics AWS Lambda EMR（Spark Streaming や Apache Fink 等を利用）

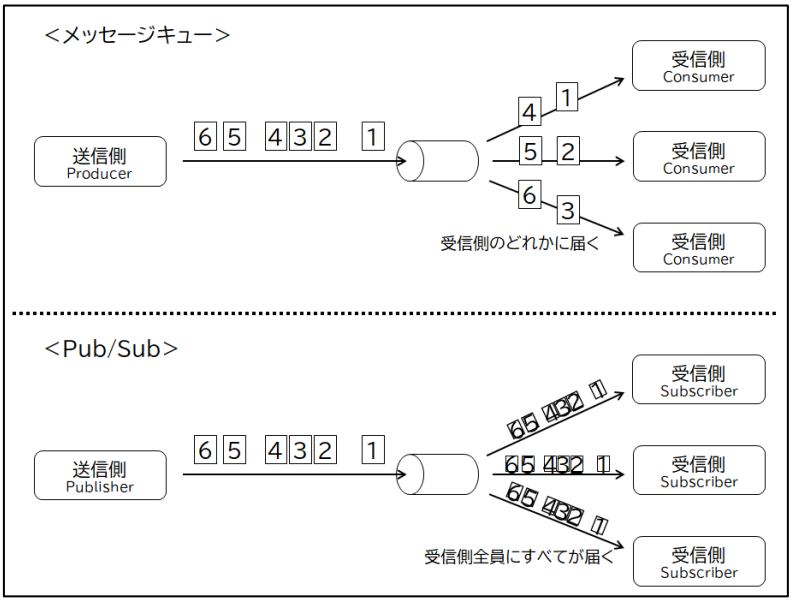
もともとストリーム処理という技術分野自体が Apache Hadoop を基盤とする OSS と共に成長してきたこともあり、ストリーム処理エンジンとしての機能を考えるとそれらに一日の長があります。EMR 上でそれらを動かす事もできますが、今回の記事ではサーバーレスありきということでそれらは対象外とします。

Kinesis Data Streams

AWS のストリーム処理サービスにおいて一番重要なものが、メッセージキューを提供する Amazon Kinesis Data Streams です。まず Kinesis Data Streams が中心となり、そこに様々なデータ発生源が集まり、用途に応じたストリーム処理エンジンに流し込みます。

Kinesis Data Streams は、大量のストリームデータ扱うことができる Pub/Sub 型の非同期メッセージングサービスです。混同されがちな一般的なメッセージキュー（Producer/Consumer 型）では、送信者（Producer）から送られたメッセージは、「どれか一つ」の受信者（Consumer）に届きます。Kinesis Data Streams を含む Pub/Sub では、送信者（Publisher）から送られたメッセージの全てが、全ての受信者に届きます。

ストリーム処理とは何か？＋2016 年の出来事
https://www.slideshare.net/SotaroKimura/ss-72769963
最近のストリーム処理事情振り返り



またデータの欠損・重複については「at least once（少なくとも 1 回）」を提供しているため、基本的にデータが「抜ける」ことはありませんが、同じデータが二回以上届く事があります。そのため受信者は重複したデータが届いても良いような処理（冪等性）にするか、それによって発生する誤差を諦める必要があります。

もう一つの特徴は、受信者が非同期であるということです。言い換えれば Pull 型の Pub/Sub です。受信者のアプリケーションが Kinesis Data Streams に対してポーリングを行い新しいデータを取得します。1 秒あたり 5 回にポーリングが制限されているため、200 ミリ秒より細かい単位でデータを処理したい場合には Kinesis Data Streams を利用することはできません。

非同期という事は、必要な受信者にデータが届くまで Pub/Sub 側でデータを保持し続ける必要があります。Kinesis Data Streams では、受信したストリームデータを最低 24 時間その内部で保存します。それぞれの受信者は、保存データ内のカーソルとも言うべきシャードイテレーターを持ち、新着分までの差分データを取得します。

Kinesis Data Analytics

Amazon Kinesis 一族にはストリーム処理エンジンが用意されています。それが Kinesis Data Analytics です。ストリームデータに対して SQL で様々な集計を行うことに長けています。

最も分かりやすいユースケースがウインドウ集計です。ウインドウ集計では、「直近の 10 分間」などを対象範囲（ウインドウ）として集約関数を適用します。さらに、集計の間隔によって 3 種類に分けられます。

- Tumbling Window 集計：一定時間毎にその間の結果を返します。
  例:10 分ごとに、その 10 分間の間を対象に集計
- Hopping Window 集計:一定時間毎に、直近の期間に対する集計を返します。
  例：1 分ごとに、直近 10 分間の間を対象に主系
- Sliding Window 集計：イベントのタイミングで、直近の期間に対する集計を返します。集計するタイミングが異なるのみなので、Hopping Window とあわせて Sliding Window と呼ぶことも多いです。
  例：ログインしたときに、直近 1 時間での不正ログインの回数を数える。

株価情報を模したデモ用ストリームデータ[31]が用意さているため、これを使って Tumbling Window 集計と Sliding Window 集計を行ったときの SQL を紹介します。なお、これらは全て Developer Guide[32]にあるものです。

■ Tumbling Window

60 秒毎に、銘柄毎の安値・高値を出力するのが以下の SQL です。

```
CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM" (
                                  ticker_symbol VARCHAR(4),
                                  Min_Price     DOUBLE,
                                  Max_Price     DOUBLE);

CREATE OR REPLACE PUMP "STREAM_PUMP" AS
  INSERT INTO "DESTINATION_SQL_STREAM"
    SELECT STREAM Ticker_Symbol,
                  MIN(Price) AS Min_Price,
                  MAX(Price) AS Max_Price
    FROM    "SOURCE_SQL_STREAM_001"
    GROUP BY Ticker_Symbol,
             STEP("SOURCE_SQL_STREAM_001".ROWTIME BY INTERVAL '60' SECOND);
```

前半では出力先のストリーム「DESTINATION_SQL_STREAM」を定義しています。後半で肝心のストリーム処理を定義しています。

重要となるのは、GROUP BY 句に書かれている以下の部分です。ROWTIME は Kinesis Data Analytics がデータを受け取った時刻（処理を行う時刻とほぼ同じ）ですので、60 秒毎にこの SQL がその期間を対象に実行されることを指示しています。

```
    GROUP BY Ticker_Symbol,
             STEP("SOURCE_SQL_STREAM_001".ROWTIME BY INTERVAL '60' SECOND);
```

---

[31] 株式銘柄コード（4 文字）、業界分野、差分、株価の 4 項目がひたすら流れてきます。

[32] https://docs.aws.amazon.com/kinesisanalytics/latest/dev/windowed-sql.html

■ Sliding Window

1 件データが来る度に、直近 1 分間の安値・高値・平均値を出力するのが以下の SQL です。

```
CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM" (
                         ticker_symbol VARCHAR(10),
                         Min_Price     double,
                         Max_Price     double,
                         Avg_Price     double);

CREATE OR REPLACE PUMP "STREAM_PUMP" AS
   INSERT INTO "DESTINATION_SQL_STREAM"
     SELECT STREAM ticker_symbol,
                   MIN(Price) OVER W1 AS Min_Price,
                   MAX(Price) OVER W1 AS Max_Price,
                   AVG(Price) OVER W1 AS Avg_Price
     FROM   "SOURCE_SQL_STREAM_001"
     WINDOW W1 AS (
        PARTITION BY ticker_symbol
        RANGE INTERVAL '1' MINUTE PRECEDING);
```

GROUP BY 句の代わりに WINDOW 句が増えました。これが Sliding Window を定義しています。1 分間のウインドウを W1 として定義し、「MIN(Price) OVER W1」のように集約関数の対象期間として指定しています。

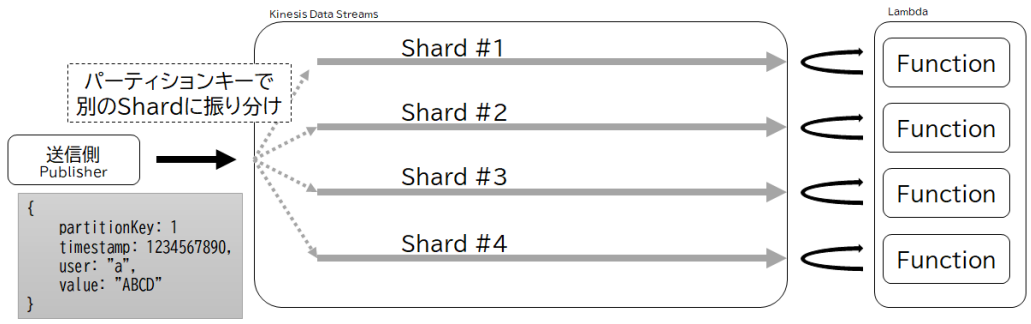
このように、流れてくるデータに対して過去の直近のデータを利用した集計などを簡単に実現できるのが Kinesis Data Analytics です。提供されている SQL で表現できないことはできませんが、Lambda を併用した前処理など機能追加によってできることが増えていくのは間違いないでしょう。

Kinesis Data Streams と Lambda

今のところ AWS からフルマネージドサービスとして提供されているストリーム処理エンジンは Kinesis Data Analytics のみなので、もう少し凝った処理が必要であれば Lambda を組み合わせるか、Spark Streaming や Apache Fink などを EMR 上で動かす事になります。本書では Lambda を取り上げていきます。

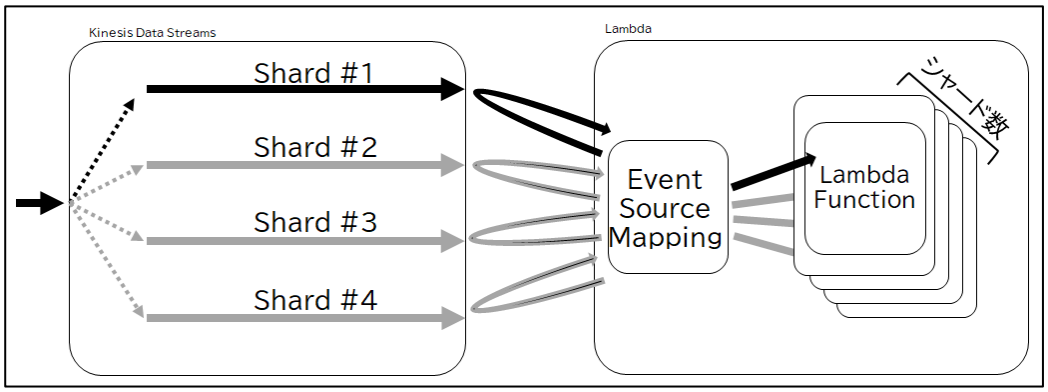
AWS におけるストリーム処理の基本は Kinesis Data Streams なので、その先に Lambda を接続することになります。Kinesis Data Analytics ではあまり気にする必要がありませんでしたが、Lambda と組み合わせる場合は Kinesis Data Streams の仕組みをもう少し深く知る必要があります。

最初にも紹介したとおり Kinesis Data Streams は Pull 型かつ Pub/Sub 型のメッセージングサービスですが、さらに付け加えるなら大量のデータを処理できるようにスケーラブルな分散システムとして設計されています。一つのストリームは、内部では「シャード（Shard）」という単位で分割されて処理されます。Kinesis Data Streams に送られたデータをどのシャードに割り当てるかを決めるのに使われるのが「パーティションキー」です。



ストリームデータはパーティションキーによってどれかのシャードに割り当てられます。同じシャードに追加されたデータ同士はその順序が維持されますが、別のシャードに振り分けられてしまったデータは順序が解らないため、それが必要であればデータの発生時刻やシステムへの到着時刻などをデータ内に保持する必要があります。

データを取得する受信者(Subscriber)側、今回であれば Lambda の Function は、シャードの数だけ並行してデータを受信する必要があります。Lambda の場合は、接続された Kinesis Data Streams のシャードの数だけ Function が同時実行[33]され、それぞれのシャードからのデータが渡されます。この処理を実際に行うのが「Event Source Mapping」です。



Event Source Mapping は、それぞれのシャードに対応するシャードイテレーターを保持し、各シャードイテレーターによって受信した新しいイベントを引数に、各シャードに対応する Lambda Function を実行します。基本的には 1 秒ごとにポーリングしますが、BatchSize というパラメータに

[33] 実際には（おそらく高速化などのため）、シャード数よりも多くの Lambda Function が同時に「起動」されているようですが、ハンドラ関数が同時に実行されるのはシャード数までとなります。

よって一度に Lambda に渡すイベントの個数の最大値を指定でき、それ以上のイベントが届いた場合はリトライされます。

Lambda Function には第 1 引数の event として引き渡されます。event.Records に配列として複数のイベントが並んでいますが、さらにその中の kinesis.data に実際のイベントデータが Base64 エンコードされた JSON[34]として格納されています。

```
{
    "Records": [
        {
            "kinesis": {
                "kinesisSchemaVersion": "1.0",
                "partitionKey": "PartitionKey",
                "sequenceNumber": "〜〜〜〜〜〜",
                "data": "eyJUSUNLR〜〜〜〜〜〜",
                "approximateArrivalTimestamp": 1538917978.966
            },
            (省略)
        },
        ...
    ]
}
```

というわけで、だいたい処理はこのような感じになります。

```
exports.handler = function(event, context, callback) {
    event.Records.forEach(function(record) {
        var message = new Buffer(record.kinesis.data, 'base64').toString('ascii');
        console.log('message:', message);
        // message を使った実際の処理
    });
    callback();
};
```

あとは、煮たり焼いたり好きなようにできます。

[34] みんなだいすき eyJ からはじまる文字列ですよ！

Azure におけるストリーム処理

Azure においても、全体像はだいたい AWS と同じです。

Event Hubs

AWS の Kinesis Data Streams に相当するのが、Azure Event Hubs です。

パーティションキーによって分散するところなど、この二つはほとんど同じような動作をしますが、いくつかの違いがあります。

- Kinesis Data Streams は、「ストリームの名前」という1階層で識別しますが、Azure Event Hubs では「Event Hubs 名前空間」と「Event Hub」という2階層で管理します。アクセス権限や性能などは「Event Hubs 名前空間」という大きな単位で管理します。
- イテレーターの管理は、Kinesis Data Streams ではクライアント側がシャードイテレーターを管理しますが、Azure Event Hubs 側にコンシューマーグループという管理単位が存在します。

Stream Analytics

これも AWS の Kinesis Data Analytics に相当し、SQL でストリームデータの集計ができます。ビルトイン関数などを見ると、本書執筆時点では Azure Stream Analytics のほうが機能が揃っているようです。

Event Hubs と Azure Functions

Pub/Sub メッセージングサービスと FaaS を組み合わせて使うときの仕組みも、おおよそ Kinesis Data Streams＋Lambda の場合とほぼ同じようにできます。

先ほども書いたとおりクライアントがシャードイテレーターを管理するのでは無く、Event Hubs 側でコンシューマーグループという枠組みでイテレーターを管理している点は注意が必要です。Lambda の感覚で、デフォルトのコンシューマーグループに複数の Function をぶら下げてしまうと、どれかの Function だけにイベントが行ってしまい、他の Function にイベントが渡らないことになります。

AWS における Event Source Mapping に代わるものとして、Azure Functions にはより汎用的なバインディングという仕組みが用意されています。管理ポータルからトリガーとして Event Hubs を指定することで、接続に必要なキーを環境変数に保存し、取得先の Event Hubs 名前空間や Event Hub の名前を function.json に保存してくれます。

ここまで設定してしまえば、あとは Lambda と同じように Function の引数として引き渡されます。Kinesis の元の仕様を引っ張っていて Base64 デコードが必要な Lambda と異なり、シンプルにイベント配列として渡されます。

```
module.exports = function (context, eventHubMessages) {
    eventHubMessages.forEach((message, index) => {
        console.log('message:', message);
        // message を使った実際の処理
    });
    context.done();
};
```

あとは、煮たり焼いたり好きなようにできます。

また、バインディングの仕組みを活用して、トリガーで入ってきたイベントに含まれるデータを利用して、CosmosDB などから検索した結果を入力バインディングとして、Function の引数に持ってきたりすることも可能です。

Databricks

Azure には、Apache Spark のフルマネージドサービスとして Azure Databricks が提供されています。本書のスコープからは外れますが、SQL で表現できないような複雑な集計でも、Spark Streaming で実装しフルマネージドなサービスの上で実行することができます。

はまりどころ

ここからはサーバーレスでストリーム処理を実装する上でのはまりどころを書いていきます。

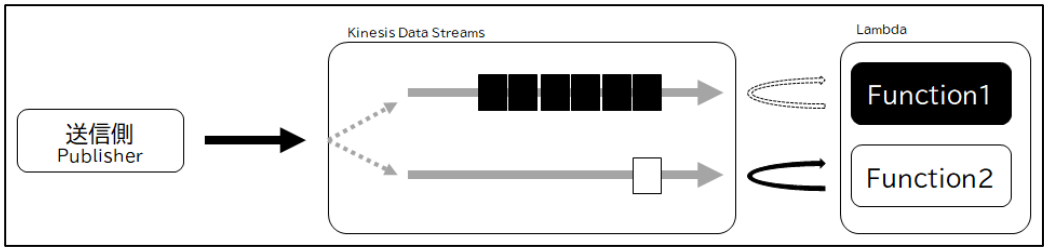
パーティションキーの選び方

Amazon Kinesis Data Streams にしろ Azure Event Hubs にしろ、パーティションキーによって内部で複数のシャードに分散され、さらにシャードを単位として後段の処理が呼び出されます。そのためパーティションキーをどのように選択するかが重要となります。

うまくパーティションキーが設定できていないと、このようなことが発生します。

たとえば、2つのシャードで動かしていた処理があるとします。何らかの理由で Function の処理に時間が掛かるようになっていくと、送信側から入ってくるペースよりも処理できるペースの方が遅くなってしまい、そうなるとストリームの中に未処理のイベントが溜まっていくようになります。

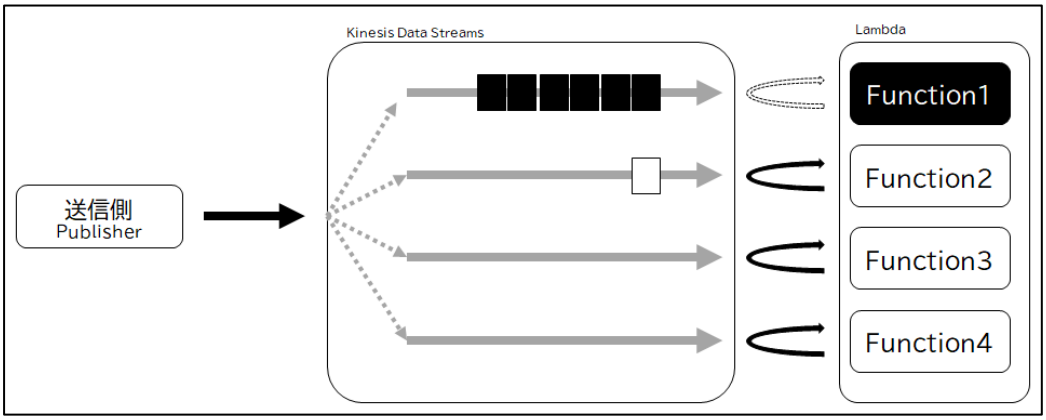
あくまでピーク時だけイベントが多いような場合であれば、ピークを超せばだんだん「処理が追いつく」ようになります。慢性的に処理能力が足りない場合はどんどんストリーム内で未処理のイベントが増え続け、設定された保存期限（デフォルトはどちらも24時間）を超えると古い順にイベントが捨てられます。これは Kinesis Data Streams なり Event Hubs じたいが持つ読み書き量の制限とは別に発生します。



Function の同時実行数を増やすためには、シャード数を増やす必要があります。

ところが、特定のパーティションキーにデータが偏っていると、せっかくシャード数を増やしても同じシャードに振り分けられ、特定の Function のインスタンスにデータが送られてしまい、問題が解決しません。要するに、いわゆる分散 DB[35]における鉄則とまったく同じように、「ホットキー」を避ける必要があります。

[35] そもそも、Pub/Sub メッセージングサービスというもの自体が、決められた時間で自動で削除され、特定の処理・検索だけが行えるデータベースと考えることもできます。



たとえばアクセスログの分析であれば、単純にアクセスされたパスをパーティションキーにしてしまうと、一般的にはトップページのページビューが圧倒的に多いはずで、トップページが振り分けられたシャードと後段の処理ばかり偏ってしまうわけです。

これはメッセージングサービスを境界として外部のサービスと連携する場合にも問題となります。

IoT プラットフォームの SORACOM には、IoT デバイスから UDP や TCP、HTTP などで送られたイベントを、利用者側のメッセージングサービス [36]に送り込んでくれる SORACOM Funnel というサービスがあります。もともと SORACOM Funnel は大量のデバイスからのデータを集めて送り込んでくれるという設計思想ということで、同じデバイスからのイベントの時系列を保存するためにパーティションキーとしてデバイスを識別できる IMSI[37]を利用していました。

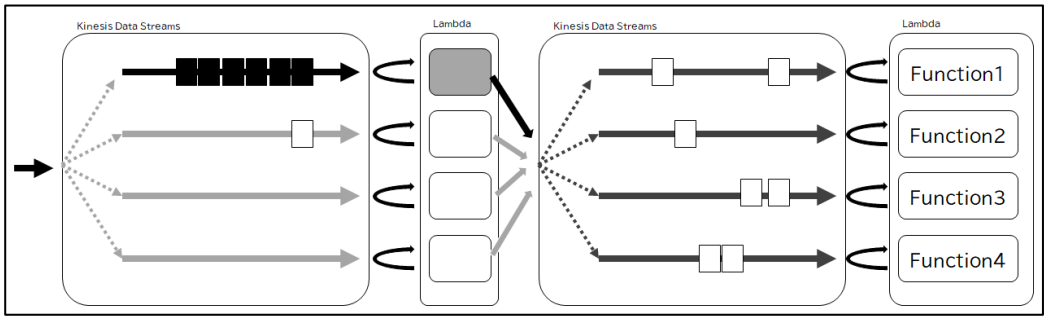
この SORACOM Funnel を利用して、少数のデバイスから大量のイベントを送信するようなシステムを作ったときに問題が起こりました。Kinesis Data Streams に送られたデータの処理が追いつかなくなったときに、シャード数をいくら増やしてもデバイス数までしかイベントが分散されず、また特定のデバイスからのデータが特に多かったことから、上記とまったく同じ状況に陥りました。

回避策として、重い処理を引き受ける別のストリームを作成し、SORACOM Funnel が投げ込むストリームからは別ストリームにパーティションキーを付け替えて投げ込むだけの軽量な処理だけをぶらさげました。

---

[36] Amazon Kinesis Data Streams、Azure Event Hubs だけでなく、Google Cloud Pub/Sub、AWS　IoT や、ウイングアーク 1st 社 MotionBoard など様々なサービスに送信することができ、投げ込む際の認証情報なども管理してくれます。SORACOM Funnel について詳しくは次の章で！

[37] SIM 1 枚ごとに割り当てられている識別子



また、これと並行して SORACOM 社に要望を出し、IMSI の代わりにランダム値をパーティションキーとして使用するオプションが実装されることになりました。もちろん、これは同じデバイスからのイベントの順序関係が維持されなくなるというデメリットを抱えています。このケースではそれが許容できるものだったため、最終的にランダム値を利用するようになっています。

このように、メッセージングサービスを介して外部連携を行うパターンが、サーバーレスの普及と活用によって今後増えて行くと思われますが、そういった場合にはパーティションキーの選択を双方で上手く合意する必要があります。

監視

システム運用において監視は義務です。ですが、ストリーム処理の場合には、リアルタイム処理という見た目に反してほとんど全てのものが非同期システムの連携として動いているため、監視のポイントも従来のような同期型のシステムとは変わってきます。

結論から言ってしまうと、以下を監視すれば良さそうです。

- ストリームに溜まっている未処理イベントの遅延時間

これが最優先監視対象です。

Amazon Kinesis Data Streams であれば IteratorAgeMilliseconds というそのままずばりのメトリクス [38]が存在します。これは、ストリームから取り出した最後のイベントで、そのイベントがストリームに書き込まれた時刻と現在時刻の差分です。言い換えれば、イベントがストリームに入ってから出るまでの時間です。1秒毎に Lambda からポーリングされるため、滞りなく処理できている場合には1秒前後を推移します。

[38] ストリーム側にも、Lambda の Event Source Mapping 側にも、どちらも存在します。

Azure Event Hubs には今のところそのようなメトリクスが用意されていないため、あらかじめイベントに発生時刻（あるいはクラウドへの到着時刻）を含めておき、Event Hubs から取り出した Function 側でイベントの発生時刻と現在時刻を比較する必要があります。

これが増加している場合、他のメトリクスを使って何が起きているかを判断していく必要があります。

■ ストリームの流量

そもそもストリームデータの量が増えているのであれば、それを把握する必要があります。これは AWS も Azure どちらも普通に提供されているので素直に監視しましょう。

■ Function の処理時間

入ってくるストリームデータの量に大差ないのに処理待ちイベントが増えているのであれば、後段の Function の処理に時間が掛かっている可能性が高いです。

Function の平均実行時間だけでなく、最大値や 99%値、95%値なども見ると、「ほとんどは早く終わるけどたまに時間が掛かるイベントがある」「そもそもたまに失敗してタイムアウトまで掛かっている」などが判明します。

■ 一度に取り出したイベントの数

処理時間と合わせて見ておきたいのが、1 回のポーリングで取得したイベントの数です。

AWS も Azure もその最大値を設定できますが、不用意に低かったり高かったりすると割り当てられている CPU やメモリの効率が悪く性能を十分に発揮しきれないことがあります。そのため、Function に一度に渡されるイベントの数を上手く調整してあげることで、シャード数を増やさず [39]に処理性能を引き上げることが可能です。

[39] なにげに Kinesis Data Streams も 1 シャードあたり月 1500 円ほどと安くありません。

At least once と向き合う

基本的に Pub/Sub メッセージングサービスは「At least once（少なくとも 1 回）」というモデルを採用しています。既に解説したとおり、読んで字のごとくですが、一つのイベントが少なくとも 1 回配信されます。

同じデータが重複することがある、ということを十分に設計に組み込む必要があります。

たとえば、イベント自体にランダムや連番などユニークとなるキーを含めておき、最終的に集計する直前でキャッシュ系データストアなどによって重複削除したり、より単純に連番で同じか古いものを無視したりといった対応が考えられます。

また Elasticsearch であればドキュメント ID を使うことで、同じデータが何度保存されてもデータが重複せず上書きされるようになります。DynamoDB 等を使う場合も同じようにキー設計をすることが可能です。

逆に同じデータが複数回来てもサービス上問題がない用途であれば、データベースへの書き込み量などは増えたままになりますがそのまま気にせず処理してしまうという選択肢もあります。

このように「At least once」をデータ処理全体の基本的な設計に組み込む必要があります。

SORACOM 買収からわかる IoT の未来

本書の執筆中に「株式会社ソラコムの KDDI グループ参画」というニュースが飛び込んできました。日経新聞曰く 200 億円という買収額をどう感じるかは視点次第ではありますが、玉川さんのインタビュー[40]にある「ビジョンをやりきるため、アクセルを踏み込むための M&A」という言葉のとおり、世界をねらうためのステップとして期待をしています。

オープニングポエムはさておき、単なる IoT/M2M 機器向けに価格プランを作り込んだだけの安価な MVNO キャリアと勘違いされがちな SORACOM ですが、その真価は「通信とクラウドを融合した IoT 通信プラットフォーム」であることです。本章では、これがどういうことか解き明かしていこうと思います。

SORACOM とは

端的に言えば、SORACOM は株式会社ソラコムが提供する IoT 通信プラットフォームです。

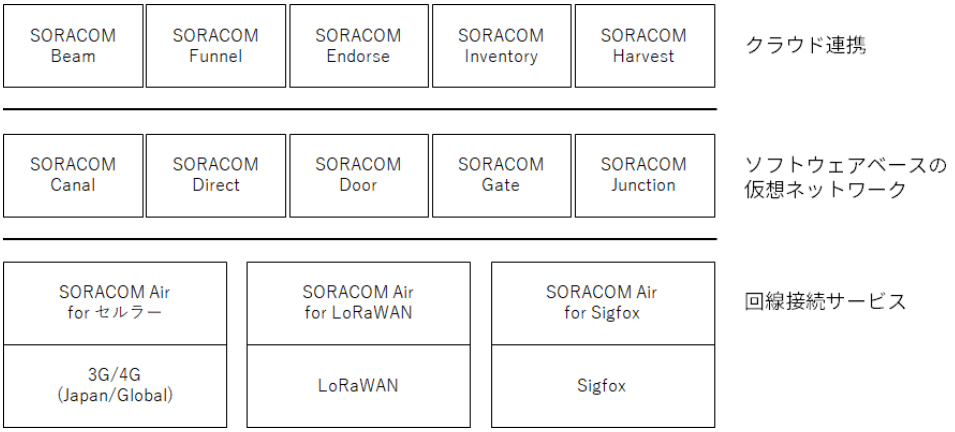
正直「IoT なんちゃらプラットフォーム」という単語自体にバズワード感がありますが、一般的には IoT[41]デバイスが通信をするために必要なものを揃えたパッケージという感じで使われているようです。SORACOM のように通信にフォーカスしたものだけでなく、（ネットワーク接続を前提に）IoT デバイスから集まってきたデータを分析・可視化するサービスや、デバイスからデータをクラウドに投げるときのライブラリ群、あるいはそれらを包括的に提供するものといくつかのパターンがあるようです。

SORACOM はざっくり三つの役割を提供しています。

---

[40] ソラコム玉川社長に聞いた「KDDI 入り」の背景とソラコムのメッセージ
http://ascii.jp/elem/000/001/525/1525188/

[41] IoT の定義については、Internet の指す対象や CPS（Cyber-Physical System）との違いなど議論がありますが、本書では「大量のデバイスを IP ネットワークに接続して何かする」ぐらいの緩い定義とします。



SORACOM はこれらを下から上まで「垂直統合」で提供し相互に価値を高め合うことで、利用者に対して高度なサービスを提供しているわけです。

回線接続サービス

IoT デバイスがネットワークに接続するための回線を提供する、プラットフォームとしては一番低い部分です。サービス名称としては「SORACOM Air」が対応し、SORACOM の最初のサービスでもあります。

当初は NTT ドコモの MVNO として日本国内向けにサービスを開始しましたが、その後海外でも利用可能なグローバル向け Air SIM が提供されたほか、920MHz 帯の LPWA（Low Power Wide Area Network[42]）である LoRaWAN と Sigfox にも対応しました。（追記：その後、KDDI 回線を利用できる SORACOM Air SIM（plan-K）と、今回の SORACOM Button も利用している LTE-M（plan-KM1）が提供されました。）

また、株式会社ソラコムとしてのサービスではありませんが、ソラコムが開発した SORACOM のエンジンである「SORACOM vConnec Core」を KDDI が採用した「KDDI IoT コネクト Air」があります。MVNO と同等の接続サービスとしては同じように提供されていますが、この後に紹介する上位層の機能は残念ながら SORACOM 本家から若干遅れての提供になっているようですが。KDDI グループへの参加により、サービス内容の共通化も進んでいくと考えられます。なお、技術的詳細は明かされていないので、KDDI としてのサービスについて、これ以降は触れません。

[42] 間欠的に通信を行うことで、低電力（設定によっては乾電池で数ヶ月以上）で広域（数 km）の通信を実現しようとする無線通信のカテゴリ

（上記のとおり SORACOM 本体から plan-K が提供されたことにより、こちらの KDDI IoT コネクト Air の新規申し込みは終了になった模様です。その結果、全てのサービスがドコモ回線と同じスピードで提供されるようになっています！）

SORACOM Air は NTT ドコモと L2 卸契約する MVNO ですが、NTT ドコモからの専用線から先の MVNO キャリアとして必要なシステム全てを AWS 上で構築しています。データトラフィックの転送や通信網の管理を、クラウドネイティブな設計でソフトウェアによって実装しています。これにより、利用数の増加を見込んだ設備投資を避けつつ、高い頻度での開発サイクルを回しています。その副産物として、1 日 10 円＋最低 0.2 円/1MB（たとえば月間 1GB であれば約 500 円）という低トラフィックでは極めて安価な従量課金を実現しています。

ただ、安いことは SORACOM の価値の中ではぶっちゃけ些細なことではないでしょうか。単に安いインターネット接続回線だけが欲しいのであれば、月額 300～500 円や、一定転送量まで 0 円という MVNO キャリアも登場している昨今です。

SORACOM Air の価値の半分は、その管理機能を API として提供していることにあります。API から SIM ごとの設定（回線速度や回線自体の有効・無効など）を変更でき、通信量などを監視して AWS Lambda を呼び出すイベントハンドラー機能もあります。これを利用することで、SORACOM の上に仮想 MVNO 事業者を構築したり、SIM を組み込んだ形で製品を出荷してから契約状況に合わせて通信を有効化・無効化したりといったことが可能になるわけです。

残りの半分は、SORACOM Air という回線契約があることで、安全にデバイスを識別でき、それによって上位層の機能を提供できるということです。セルラー[43]であれば SIM が、Sigfox であればデバイス内蔵のセキュアエレメントが耐タンパー性 [44]を持ち、デバイス識別のための秘密情報が安全に保存されています。また LoRaWAN の場合にも、出荷時に SORACOM 側と同期されてデバイスに設定される AppKey を使ってセッション鍵を作成するため、耐タンパー性とまではいきませんがそれなりに保護されています。

接続回線という一番下の層において個別のデバイスを一つの ID 基盤上で識別できているからこそ、異なる通信網を性質の違いのみを意識するだけで、その上に多種多様なサービスを統一的に提供できます。

概念的な表現をすると、閉域網の中でデバイス別に IP アドレスが割り振られる旧来のネットワークセキュリティのモデルに対して、回線の接続点で直接認証される ID ベースのセキュリティという新しいモデルがあり、それに基づいて SORACOM のあらゆるサービスが実現されています。

---

[43] いわゆる 3G/4G など携帯電話回線ベースの技術

[44] 壊しても中身の暗号鍵などを見られない性質

要するに、「Air」に続く「B」以降のサービスのために、SORACOM Air としては様々な通信方式に対応する必要があり、今後来る 5G ベースの携帯通信網や、グローバルの世界制覇のために KDDI と手を組んだのだと思います。

ソフトウェアベースの高機能な仮想ネットワーク

先ほど書いたとおり、SORACOM のネットワークは全てクラウド上のソフトウェアで構築されています。つまり一種の Software-defined Network（SDN）や Network Function Virtualization（NFV）です。通信回線の反対側にそのまま SDN/NFV の世界を接続したことで、閉域網として様々な機能を提供しています。

閉域網という視点では接続先に応じて 3 種類のサービスを提供しています。AWS 上の自分の VPC に接続する最も安価な「SORACOM Canal」、物理専用線として AWS Direct Connect を利用する「SORACOM Direct」、インターネット VPN を利用する「SORACOM Door」です。

これらを利用する場合は、Virtual Private Gateway（VPG）という仮想ルータを構築して、SORACOM Air の回線ごとに VPG に接続します。

自分独自の VPG を利用することで、デバイスに割り振られる IP アドレスレンジ [45]を変更したり、デバイス毎に固定の IP アドレスを指定したりといったことも可能です（次に紹介する SORACOM Gate を使わない限りその IP アドレスが直接見えることはありません）。ただし、これらは閉域網と言っても VPG で NAPT されるため、あくまでデバイス側からクラウド側への方向の接続のみを提供しています。

クラウド側からデバイス側への接続、あるいはデバイス同士の接続を実現するのが「SORACOM Gate」です。SORACOM Gate を有効にするとデバイス毎の IP アドレスを利用して相互に通信が可能となります。さらに、SORACOM Canal 等で接続されたネットワーク上に Gate Peer と呼ばれるルータ（実際には VXLAN が設定された Linux ホスト等）を構築することで、そこからデバイスまで一つの仮想 L2 ネットワークが提供され、自分のネットワークからデバイスに直接接続できるようになります。Gate Peer まで L2 で伸びてきてくれるため、そこから自分の VPC 内のネットワークにさらに L2 ブリッジしても良いですし、もっとシンプルに NAPT するのも手軽です。

さて、このあたりまでは気合いの入った閉域網サービスであれば実現できるところですが、SDN であることを最大限に活かす SORACOM らしいサービス「SORACOM Junction」がつい先日リリースされました。これは VPG を通過するパケットに対して、ミラーリング、リダイレクション、インスペクションの 3 つの操作を提供します。

---

[45] デフォルトでは 10.128.0.0/9

ミラーリングとリダイレクションは、名前の通りパケットを複製したり転送先を変更したりすることで、ユーザ企業が独自の監視システムを通したりトラフィックの制御を行ったりと言うことが可能となります。インスペクションはパケットの統計情報を外部サービス[46]に送信します。

これら SORACOM の高機能な仮想ネットワークは、全てクラウドネイティブな設計で実現されているため、SORACOM の利用者やその転送速度・秒間パケット数などが増えても（万が一 AWS のリソースが頭打ちしない限り）いくらでもスケールすることができます。その一方で、たとえば SORACOM Canal の VPG は 1 時間あたり 50 円（30 日間で 36,000 円）と、きちんと利用者にそれなりの負担が求められます[47]。このあたりはうまく技術上の制約をビジネスとして成立させていると感じます。

モバイル通信網の課金体系は、行き過ぎたインセンティブなどにより極めて歪になってしまっていますが、SORACOM の素直な課金体系を見るとほっとします。

クラウド連携

SORACOM の真価はクラウドとの統合にあります。IoT でビジネスをするために必要なのは、単に IoT デバイスを IP ネットワークに接続することだけではなく、その IP ネットワークを介してクラウドの向こう側に大量のデータを期待する品質で届けることです。通信とクラウドの融合を謳っているように、SORACOM には IoT デバイスをクラウドと連携するための仕組みが数多く用意されています。

SORACOM のクラウド連携機能は、大きく二つに区別できます。一つは、AWS や Azure、GCP、あるいは独自に構築したサーバなどの、「クラウド側」との橋渡しをする機能です。サービスとしては「SORACOM Beam」と「SORACOM Funnel」があります。単純にデータを左から右へと転送するだけで無く、クラウドに送る際の認証や暗号化処理、データ量の限られた LPWA を利用する時のデータ形式変換など、「IoT デバイスがクラウドと通信する際のあるある処理」を SORACOM が肩代わりしてくれます。

もう一つは SORACOM 自身がクラウドとして直接的な機能を提供するもので、「SORACOM Endorse」「SORACOM Harvest」「SORACOM Inventory」が対応します。SORACOM Harvest のように単体で完結するものもあれば、SORACOM Endorse のように別の枠組みと連携するための機能もあります。ここからは、これらを個別に掘り下げていきます。

---

[46] 送信先にできるクラウドサービスはこの後紹介する SORACOM Funnel と同じようです。内部的には同じ枠組みに載っていると思われます。

[47] 月 36,000 円で用意できる AWS のインフラ環境は何だろう……と考えていくと、VPG の仕組みが妄想できそうな気がしてきます。

SORACOM の世界観

SORACOM のクラウド連携について具体的な機能を紹介する前に、まず SORACOM が IoT プラットフォームとして提供しようとしている世界観について触れておきます。

IoT における代表的な脆弱性を OWASP がまとめているので、以下に引用します [48]。

> I1 Insecure Web Interface
> 安全でない（製品の）Web インターフェース
>
> I2 Insufficient Authentication/Authorization
> 不十分な認証/認可
>
> I3 Insecure Network Services
> 安全でないネットワークサービス
>
> I4 Lack of Transport Encryption/Integrity Verification
> 転送路における暗号化・完全性確認の欠如
>
> I5 Privacy Concerns
> プライバシーに帯する懸念
>
> I6 Insecure Cloud Interface
> 安全でないクラウド上の Web インターフェース
>
> I7 Insecure Mobile Interface
> 安全でないモバイルアプリのインターフェース
>
> I8 Insufficient Security Configurability
> 不十分なセキュリティ設定
>
> I9 Insecure Software/Firmware
> 安全でないソフトウェア・ファームウェア（の更新）
>
> I10 Poor Physical Security
> 物理セキュリティが弱い

IoT デバイスが関連するのは I1、I2、I3、I4、I8、I9、I10 あたりですが、IoT 固有の物理セキュリティ（I10）を除いて、一般的なウェブアプリ等であれば当然のようにどれも考慮されているべきはずのものばかりです。これらがなぜ IoT デバイスで特に問題にされるかと言えば、IoT 固有の理由でそれらを設計時に「妥協」してしまうからです。

IoT デバイスが一般的なサーバ側システムと異なるのは大きく 2 点あります。

一つは計算機としての性能です。最近になってようやく ARM ベースなどの比較的高性能なデバイスも増えてきましたが、消費電力であったりサイズであったり、あるいは予算や部材の都合で依然として性

[48] https://www.owasp.org/index.php/Top_IoT_Vulnerabilities

能の低い IoT デバイスとはまだ何年も付き合っていく必要があります。ところがデバイス自身を正しく認証する、あるいはデバイスに接続するユーザを認証するには、公開鍵暗号やハッシュ関数などの計算が必要となり、決して低くない演算能力が必要となります。さらに大量のデータをクラウドに送信するようなデバイスでは継続的な暗号化も必要となります。

旧来のネットワーク制御デバイスでは、ネットワークセキュリティを前提として「妥協」をしてきました。その結果として、USB メモリなど様々な手法を組み合わせた Stuxnet などの被害が発生しています。何らかの手段で、低い処理性能しか持たない IoT デバイスからクラウドまでの通信を暗号学的に守っていく必要があることは確定的に明らかです。

もう一つが、IoT デバイスの数が多いということです。様々なデバイスをネットワークに接続して安全に機能させるには、それぞれのデバイスを個別にセキュアに識別する必要があります。そのためには、電子証明書あるいは事前共有鍵どちらでも良いですが、デバイス毎に何らかの秘密の認証情報を持たせる必要があります。ところが、数百数千が前提となるような IoT デバイスでは製造時に個別の情報を持たせ、それをそのまま運用時にも利用するということが難しくなってきます。したがって運用開始時に認証情報を投入する必要が出てくるわけですが、物理的な作業をどう減らすかが重要であり設計の難しさが一気に上がります。また故障時の交換なども踏まえると、開始時だけではなくシステムの運用ライフサイクル全体の中でデバイスの識別方法を設計する必要があります。

SORACOM はこの課題に対して「まず接続回線を安全に識別し、その上で SORACOM 側に処理を委譲する」という解決策を提示しています。SORACOM Air の回線は、3G/4G 網や LoRaWAN/Sigfox のそれぞれの暗号学的手段によって安全に認証・暗号化されています。そこで、IoT デバイスは純粋にやりとりしたいデータだけを SORACOM に送り、SORACOM が認証や暗号化など本来 IoT デバイス上で行うはずだった処理を肩代わりします。

また、IoT デバイスは必ずネットワークに接続する必要があり、設置時に SIM を挿したり、LoRaWAN や Sigfox のデバイス ID を登録したりします。これまでは別の手段で証明書を配布するなど認証のためだけの作業が必要でしたが、通信路に紐付いてクラウド連携を行うことでそういった作業をゼロにします。

これこそが、SORACOM の世界観において核となる考え方です [49]。

プラットフォームというものを提供するということは、そのプラットフォームの世界観――すなわち、IoT/M2M デバイスにおける通信とはどういうものなのか、運用も含めたシステム全体をどういう抽象化に基づいて IoT デバイスやクラウドを設計するのか、という共通認識――のシェアを取り合う「抽象化戦争」に参加するということです。

---

[49] 偉そうに書いていますが、一ユーザから見た勝手な決めつけです😇

この SORACOM の世界観は、IoT プラットフォームにおける象化戦争においてかなり強い魅力を持っているように思います。

さて、そろそろ個別のサービスを紹介していきます。

SORACOM Beam

SORACOM Beam は、端的に言えば高機能なアプリケーションプロキシサーバです。

SORACOM 内のエントリポイントに IoT デバイスから送りやすいプロトコルで送信すると、上手い具合にプロトコルを変換したりしてクラウド側に送信してくれます。SORACOM が提供するクラウド連携機能のなかでもっともシンプルなものですが、SORACOM が提供しようとする世界観に触れる入口として一番相応しいものです。

また、SORACOM 上で全ての SORACOM Air 回線はグループ単位で管理されますが、SORACOM Beam はこのグループに対して設定されます。したがってグループの Beam 設定を変更するだけで、そこに紐付いた全ての IoT デバイスからの送信先が一括で管理できます。当然ながら、Beam に限らずこのような設定を SORACOM API で変更することも可能であり、デバイスを含むシステムのオンライン開通などを構築することもできるでしょう。

利用するプロトコルで分類すると、6 種類の動作に分けられます。

HTTP ⇒ HTTP/HTTPS

IoT デバイスから HTTP でエントリポイントにリクエストを送信すると、そのリクエスト内容をクラウド側の URL に対して HTTP もしくは HTTPS でリクエストを転送します。クラウド側がインターネットの向こう側にある場合など、HTTPS による通信の暗号化処理を SORACOM 側にオフロードできます。IoT デバイスから SORACOM までは各通信手段による暗号化や、NTT ドコモから SORACOM までの専用線で守られているため、SORACOM までは HTTP で良いわけです。

クラウド側にリクエストを送信するときに、接続元 IoT デバイスの IMSI[50]・IMEI[51]や、事前共有鍵に基づいた電子署名、個別に指定したカスタムヘッダを HTTP リクエストヘッダに追加することができま

[50] International Mobile Subscriber Identity：SIM ごとに割り当てられる ID

[51] International Mobile Equipment Identity：3G/4G の携帯電話機・モデムごとに割り当てられる ID

す。これによって、クラウド側は「誰からの通信なのか」を識別することができます。これはこの後の他の HTTP 転送先でも共通です。

細かくは、ドメイン上の全てのパスをそのまま利用する「Web エントリポイント」と、パスごとに個別の転送先を設定できる「HTTP エントリポイント」の 2 種類が用意されています。

TCP ⇒ HTTP/HTTPS

TCP でエントリポイントに接続して生のデータ列を投げると、HTTP POST リクエストに変換してクラウドに送信します。データ列は、以下のように Base64 エンコードで JSON に埋め込まれて本文として送られます。

```
{"payload":"Base64 エンコードされたメッセージ"}
```

クラウドからのレスポンスもそれっぽく返ります。

```
400 Message from server
```

UDP ⇒ HTTP/HTTPS

TCP と同様に、受信したデータを HTTP/HTTPS に変換して送信します。

TCP ⇒ TCP/TCPS

TCP で送られた内容を、そのまま TCP でクラウドに送信します。IMSI や IMEI 等を送りたい場合は、HTTP リクエストヘッダの代わりに、先頭行として送信されます。

MQTT ⇒ MQTT/MQTTS

クラウド側の MQTT ブローカーと相互に Publish/Subscribe を転送します。

LoRaWAN ⇒ HTTP/HTTPS

LoRaWAN デバイスから送信された Uplink データを、HTTP/HTTPS でクラウドに送信します。先ほどの TCP からの変換と似たような動作をしますが、Base64 ではなく Hex 表記が使われるほか、経由した LoRaWAN ゲートウェイの情報や受信電波強度（RSSI）などの追加情報を含む JSON として送信されます。

LoRaWAN などの LPWA では転送できるデータ量が極めて少ない[52]ため、JSON のような富豪的なデータを来ることはできず、まさに「1 ビットは血の一滴」と送らなければならないデータをビット単位で詰め込んで送ることになります。それをどこかでクラウド側で扱いやすい形に変換しなくてはなりませんが、バイナリパーサー機能を利用することで、バイナリデータを指定した通りに分解して JSON 上に展開することができます。どこかで必要になる処理なので、LPWA の利用者には非常に便利と思います。バイナリパーサーはこの後紹介する SORACOM Funnel や SORACOM Harvest にも対応しています。

SORACOM Funnel

筆者が一番好きなサービスがこの SORACOM Funnel です。

SORACOM Beam が単純なアプリケーションプロキシという形態だったのとは対称的に、SORACOM Funnel は様々なクラウド固有のインターフェースに直接データを投げ込むことができる「クラウドリソースアダプター」です。

様々なパブリッククラウドはデータ収集用のサービスを提供していますが、IoT デバイスからそこに送信しようとすると、IoT デバイス上でクラウド接続用の SDK を組み込んだり、そこに食わせる認証情報の管理が必要となったりします。SORACOM の世界観の通りそれらの面倒な処理全てを肩代わりしてくれるのが SORACOM Funnel です。IoT デバイスは、送りたいデータをそのまま SORACOM Funnel のエントリポイントに HTTP もしくは TCP/UDP で送るだけです。直接 IoT デバイス上のアプリケーションから送信しても良いし、例えば Fluentd が使える環境であれば out-http プラグインが直接使えたりします。

例えば SORACOM Air で接続した IoT デバイスからこんな感じで HTTP POST で投げます。

```
$ curl -X POST http://funnel.soracom.io ¥
    -H Content-Type:application/json ¥
    -d '{"foo":"bar"}'
```

そしてこれを SORACOM Funnel の設定で AWS の Kinesis Streams に送るように設定しておくと、Kinesis Streams には以下のようなレコードが送信されます。

[52] デフォルト設定（Spreading Factor＝10）では、4.4 秒に 1 回ずつ、1 回に 11 バイトまで

```
{
   "operatorId": "OP9999999999",
   "timestamp": 1473322750825,
   "destination" : {
    "resourceUrl": "https://kinesis.us-west-2.amazonaws.com/teststream",
    "service": "kinesis",
    "provider":"aws"
   },
   "payloads": { "foo": "bar" },
   "imsi": "440000000000000"
  }
```

payloads の中に送信した JSON が埋め込まれているほか、SORACOM が受信したタイムスタンプ（Unixtime ミリ秒）や、送信に使われた SORACOM Air の IMSI などが含まれて届きます。クラウド側ではこれらのデータを元に送信元を識別して様々な処理を行います。ね、簡単でしょ？

送信先のサービスも、AWS であれば AWS IoT、Kinesis Streams、Kinesis Firehose と一通り、それ以外のパブリッククラウドも Azure Event Hubs や Google Cloud Pub/Sub などイベントストリーミング系に流し込むことができます。他にも、SORACOM の認定済みパートナー各社のサービスに接続することができる Partner Hosted Adapter が用意されています。現在はデータ分析や可視化エンジンなど国内 5 社のサービスに対応しています。SORACOM 接続の IoT デバイスから容易に送り込めるかは各サービスで支配的な要素になることが予想できるため、今後もどんどん対応サービスが増えて行くことでしょう。

IoT デバイス「あるある」として、技術上の理由やビジネス上の要件によって利用するハードウェアや OS、ディストリビューションに制約がかかることが多々あります。そのような環境の上で、SDK を直接動かす手間を省き極めて軽く標準的なプロトコルだけでクラウドとの連携ができることは、その開発速度の改善に貢献します。数多くの PoC（実証実験）をこなさなくてはならない IoT ビジネスにおいて、開発速度はビジネス上の支配的な要員となります。決して SDK を利用してクラウドに直接通信することが技術的に難しいわけでは無いですが、「そこは自分達のビジネスでは無い」わけで、単純に計算機資源をオフロードするという意味だけではなく、自らの実装量そのものを減らせることが SORACOM Funnel の魅力です。

さらにセキュリティ上の理由としても、IoT デバイスの盗難などを考慮する場合は IoT デバイス毎にアクセスキーなどの認証情報を個別に用意するか、一時的なリスクならば受容できる場合は盗難が発覚したときに即座に認証情報を一括で交換するなどの対策が必要となります。SORACOM Funnel ではクラウド側に対する認証情報を SORACOM 側で持つので、そもそも IoT デバイス上に何かを管理する必要がありません。こういった「考えることを減らす」という方向性のものが私は大好きです。

地味な副産物として、クラウド側の問題でデータ投入に失敗した場合なども、SORACOM 上のマネジメントコンソールで直近 2 週間のログが見られるので、IoT デバイスを触らずにトラブルシュートできるのも嬉しいところです。そもそも IoT デバイスなんてファイルシステムが read-only で書けるのは

tmpfs だけとか、書き込める容量が数 MB とかしかないとかザラで基本的に必要なログは外部に転送する必要があったりするわけで、とにかく自分でやることを減らすのが IoT 開発の肝です。

ここまでの 2 つは、IoT デバイスとクラウドを連携させるための橋渡しのサービスでした。残りの 3 つは、SORACOM 自身がクラウド的なサービスを提供するものです。

SORACOM Endorse

SORACOM のサービスの中でも、地味ながら面白いものの一つが SORACOM Endorse です。

繰り返し述べてきたように、SORACOM Air の 3G/4G で接続されたデバイスは 3G/4G の SIM により暗号学的に正しく認証されています。ですが、大量のデータ転送や通信の安定性を必要とする場合など、3G/4G 回線経由で送受信したくないというケースがあります。そんなとき SORACOM Endorse を使うことで、モバイル通信を経由して認証トークンを発行してもらうことができます。

認証トークンは標準的な JWT（Json Web Token）フォーマットで、SORACOM Air で接続に使われている IMSI や IMEI などのほか、任意の情報を追加することができます。また SORACOM の秘密鍵で署名されているため、そのため認証トークンを別のシステムに持っていても SORACOM の公開鍵で検証することができます。

これを使うと、単体で用意すると高価になりがちな耐タンパー性のあるセキュアエレメントによるデバイス認証を、モバイルデータ通信を経由するかどうかによらず様々な状況で利用することができます。COSR 設定も可能なので、ブラウザ等からの直接利用もできそうです。

自分のところでは Funnel/Beam で済んでしまって試せていないのですが、システムの実現可能性を広げることができるポテンシャルを感じています。（追記：この SORAOM Endorse の仕組みを上手く活用できる、SORACOM Krypton が登場しています。）

SORACOM Inventory

大量生産をする IoT デバイスにおける設定情報の管理や、稼働中のはずの IoT デバイスの状況を把握するためには、これまでは Beam などのクラウド連携機能を使って自前で設定レポジトリやテレメトリ収集システムなどを作る必要がありました。そういった部分を SORACOM が提供してくれるのが SORACOM Inventory です。技術的には標準規格 LwM2M（OMA LightweightM2M）のサーバです。

LwM2M は恥ずかしながら作者も SORACOM Inventory の登場まで知らなかったのですが要するに非同期に接続される IoT デバイス等に向けて設計された SNMP のようなものと捉えれば良い

ようです。具体的には、設定値の参照・変更や、非同期に通知されるデバイス側ステータスの変更監視、コマンドの送信などが行えるようです。

正直あまり詳しくないので深くは触れません。

SORACOM Harvest

最後に紹介する SORACOM Harvest は、SORACOM が提供するサービスの中でもっとも高い層にあるもので、IoT デバイスが SORACOM Funnel などと同じようにデータを投げると、SORACOM 上にデータを蓄積し、可視化できます。簡単な PoC であればもはや外部のパブリッククラウドを必要とせず、むしろ SORACOM 自身がパブリッククラウドとしての役割を果たしていると言えます。

投げ方とかは SORACOM Funnel と似ているので、内部的には Partner Hosted Adapter と同じように SORACOM Harvest Adapter のようなもので送り込んでいるような気がします。

ちょっとした可視化と API だけを提供するような「IoT プラットフォーム」が量産されている昨今ですし、なにより IoT デバイス側の開発に重きを置いてクラウド側はデータが見られれば良いようなプロジェクトであれば SORACOM Harvest はまさにどハマリだと思います。是非使っていきましょう。ここでも、重要なことは「自分のビジネスと関係無いところは、できるかぎり自分でやらない」ということに尽きます。

SORACOM 自身がデータを蓄積し始めたという事は、そのデータを対象とした様々な分析サービスなどを実現できる下地が揃ったと言えます。まだまだアルファベットには Z までたくさん残っているので、例えばリアルタイムな機械学習で異常検知できる SORACOM Machine Learning だとかそんな感じの手厚い系のサービスも出てくるのかもしれません。

（追記：この SORACOM Harvest で蓄積したデータを可視化して他の人に見せることができる SORACOM Lagoon がリリースされています。SORACOM Lagoon はオープンソース BI ツールの Grafana をベースにしたもので、グラフや地図などを用いたダッシュボードを作ったり、条件を指定して Slack などの Webhook にアラートを投げ込んだりすることができます。SORACOM の管理コンソールのログイン権限とは別に、共有 URL を発行してお客さんに直接見てもらうことなども可能です。）

SORACOM で気軽に遊んでみよう

散々述べてきたとおり、SORACOM の魅力は MVNO の回線に紐付いて提供される高レベルのクラウド連携機能です。従って、パブリッククラウドがそうであるように、実際に触ってみないとその面白さ、便利さを実感しづらいものです。

SORACOM のハンズオンもかなり広く実施されていますが、自分で勝手に遊んでみるのであれば、回線速度も不必要なので 3G で良いですし、古めの docomo 向け中古 USB モデムが 2000 円程度で転がっていたりします。USB モデムでは Windows/Mac のドライバインストール用に仮想 CD ドライブが仕込まれている場合がほとんどなので、あらかじめユーザ（人柱）による検証報告があるものを探すと良いです。

また、最近 LPWA のサービス圏内も拡がっているので、それを試して見るのも良いかも知れません。例えば LoRaWAN GPS トラッカーが 15,800 円で購入できるので、それでそのまま試すことができます。リリースされたばかりですが、Sigfox 対応センサ対応デバイスも 8,478 円とそこそこお手軽に手が出せそうな価格帯です。このような完結した IoT デバイスを買ってきて、SORACOM Harvest でまず可視化してみたり SORACOM Funnel でクラウドに送ってサーバーレス構成で分析してみたりすると、SORACOM が何を狙っているのか実感できると思います。

今後の SORACOM 予想

KDDI グループに参加したことで、特に SORACOM の方向性が変わるとは思いませんが、KDDI 自身の世界戦略と絡んで、グローバルなサービス展開の手厚さは変わっていくでしょう。また、KDDI が買収した SORACOM 以外とのシナジーというのはありそうに見えます。あとは、いつ SORACOM Air for 5G/NB-IoT とかそういったものがリリースされるかだけですね。（追記：NB-IoT ではありませんでしたが、LTE-M（Cat.M1）に対応したのは皆様ご存知の通りです。）

仮想ネットワークとしては、以前から要望を送っている VPG 上のファイアウォールや、透過プロキシなどに使える L4 レベルの Junction、トラフィック・フローダンプ、そんな感じで既存のネットワーク機能が素直に移植されていくものと思います。

クラウド連携としては夢が尽きないところですが、AWS であることを活かして自分で用意した Lambda との結合や、AWS IoT などデバイスツイン系のサービスの SORACOM 独自提供、スマートフォンの NFC/FeliCa 連携によるユーザ認証、先ほど書いた収集データの分析基盤など、複数の利用者が必要としそうな機能を継続的に貪欲に SORACOM-ize していくのではないかと思います。

Harvest のような単体で完結するサービスについても、管理者がコンソールから見るだけではなく、パートナーとの関係性もありますが、外部 ID 基盤と連携して一般ユーザ向けの画面を提供といった BI ツールの領域に踏み出す可能性もあります。（追記：既に書いたとおり、一般ユーザ向けの可視化画面を提供する SORACOM Lagoon がリリースされました。）

今後も SORACOM のニューリリースから目が離せなさそうです。

Re: 今後の SORACOM 予想

この節はまるまる追記です。

改めて 2018 年末として今後の SORACOM の流れを予想すると、来年 2019 年にはこのあたりが来るのではないでしょうか。

エコシステムの拡充

SORACOM Funnel の接続先などエコシステムの拡充は間違いなく進んでいくでしょう。

エッジコンピューティングとの連携

つい先日に、AWS のエッジコンピューティング基盤とのセキュアエレメント連携が発表されましたが、エッジとクラウドの通信など様々な形で連携が進んでいくと予想しています。

Global SIM の活用

Air Global の SIM は、SIM 内の Java アプレット実行をはじめとした国内キャリア(ドコモ/KDDI)の SIM にはできない様々な機能があります。

日本国内においてもそれらを活用した事例が増えるのは間違いが無いですし、それを前提としたさらなる SORACOM サービスが Krypton に続いて登場していくのではないでしょうか。たとえば SIM では認証だけを行い、他の通信路を利用して SORACOM との間に VPN を張り VPG に接続できるようなサービスなども考えられます。

高トラフィック通信への対応

今のところ「IoT 向け」として比較的低トラフィックな利用が想定されていますが、エッジだけで完結できない用途のビデオストリーミングの転送や、従業員に支給する各種端末など、より高トラフィックな通信を念頭に置いたサービス群が登場すると予想しています。

今のところ、SORACOM Funnel において Amazon Kinesis Video Streams に流し込む Funnel KVS が Limited Preview としてひっそりと姿を見せているほか、SORACOM Junction もトラフィック異常検知などの用途にも活用できます。

ガジェット強化

ちょうど LTE-M Buttn が出たところですが、そのまま LTE-M Button を入力機器とした活用事例も出てくるでしょうし、LTE-M Button の接点入力 Ver であったり、Wio LTE などをベースとした「ボイラープレート的なハードウェアとソフトウェアのテンプレートセット」が大きく拡充していきそうな気がしています。

Bluetooth mesh ってどうよ

2017 年 7 月 18 日、長いこと蕎麦屋の出前のような状況が続いていた Bluetooth mesh がついに正式発表されました。この記事では Bluetooth mesh について紹介します。

そもそも Bluetooth のおさらい

歴史も長く言葉や製品自体はありふれている Bluetooth ですが、色々と複雑な経緯があるので Bluetooth mesh の話をする前におさらいです。

Bluetooth は 2.4GHz の周波数帯を利用する無線通信技術です。この 2.4GHz という周波数帯は ISM バンドと呼ばれるものの一つで、本来は電子レンジなど無線通信以外に利用することを目的として確保されています。混信が多いという前提の上で、重要性の低い無線通信にも比較的に自由に利用することが多くの国で認められていることから、コードレス電話や無線 LAN などで活用されています。口の悪い人はこの 2.4GHz のことを「電波のゴミ溜め」などと揶揄したりするようですが、無線 LAN で 2.4GHz 帯と 5GHz 帯を比較したことがある人は実感があるのではないでしょうか。実際にコミケの会場では 2.4GHz 帯は不毛の大地です。

Bluetooth の規格は、3.0 までのもの（クラシック Bluetooth）と、4.0 以降（Bluetooth Low Energy、BLE）で基本的な仕様がまったく異なります。本当にまったくもって互換性がなく規格の方向性も異なるため、クラシック Bluetooth の最新である 3.0+HS と、4.x の両方に対応している機器もあります。どちらも、基本的には「ペアリング」で通信相手を特定して 1 対 1 のコネクション型の通信を行います。

これから紹介する Bluetooth mesh は、BLE ベースの規格です。

Bluetooth mesh の登場

これまでの Bluetooth は、先ほど書いたとおり 1 対 1 の接続を前提としていました。ところが、1 対 1 の通信を始めるときに、主に通信相手を探したりする用途で「アドバタイジング」という 1 対多の送信をする仕組みが用意されています。

このアドバタイジングのパケットを活用（転用？）したものが、いわゆる「BLE ビーコン」です。Apple が位置測位のために 2013 年に公開した iBeacon をきっかけとして、一気に普及することになりました。技術的には、大昔から存在するアクティブ型の RFID などと同じです。本来の Bluetooth の 1 対 1 の通信の場合は見通しでも 20～30 メートルぐらいが距離の限界と言われていましたが、短いデータを一方的に繰り返し送信する BLE ビーコンでは見通し 100 メートルぐらいは届いたりします。

これに目を付けた会社がいくつかあり、Zigbee などで既に研究が進んでいた無線メッシュネットワークの技術を BLE アドバタイジングに転用することにしました。そこから紆余曲折あった結果、Bluetooth の高音質オーディオ CODEC「aptX」などの技術をもつ CSR 社（現在は Qualcomm 社が買収）が開発した CSRmesh プロトコルをベースとする形で、Bluetooth mesh として標準規格になりました。

Bluetooth mesh の特徴

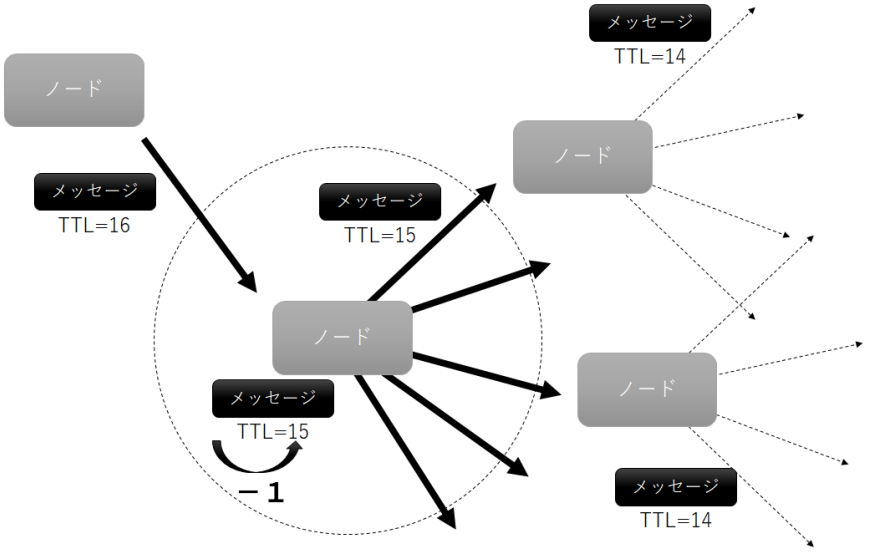
まず一番の基本的な所として、IPv4/IPv6 など他のネットワークプロトコルとの互換性はありません。まったく独自にアドレス体系、ルーティング方式、データフォーマットを持ちます。Bluetooth 4.2 で 6LoWPAN と呼ばれる方式で IPv6 の通信が可能になりましたが、それはあくまでペアリングして 1 対 1 の通信を行うためのもので、この Bluetooth mesh とは全くの無関係です。また、2016 年に発表された Bluetooth 5 とも直接は無関係で、Bluetooth mesh 自体は 4.0 以降の全てと互換性があります。

Bluetooth mesh は、数台から数千台というレンジをカバーするネットワーク規格です。したがって、インターネットのようなグローバルなものではなく、組織や建物など比較的小さいエリアをカバーするためのものです。アドレスは 15 ビット分の長さがあるので、規格上は「3 万 2000 ノードまでサポート」という表現がされているようです。後述するように、数万ノードが相互に大量の通信を継続して行うには向いておらず、用途によりけりというところはあります。

メッシュネットワークということで、ルーティングの仕方が一番気になるところだと思います。基本的な考え方は「Managed Flooding」と呼ばれ、いわゆるフラッディング型と呼ばれる方式の一種です。IP ネットワークをご存知の方は「一体何なんだ！その乱暴なネットワークは！」と驚くような手法ですが、以下の組み合わせで動いています。なお、Bluetooth mesh ではデータの単位をパケットではなくメッセージと呼びますが、だいたい同じです。

1．誰がどこに居るか気にしない⇒TTL が尽きるまでリレー

IP ネットワークで言うような「経路表」のようなものはもちません。とにかく自分宛でないメッセージを受信したら中継して再送信します。そのときに、メッセージ内の TTL を一つ減らします。TTL の考え方は IP と同じで、最初に送信するときに設定できます。



また、電波ですので指向性とかも気にせず、とにかく TTL が尽きるところまで送られます。TTL が余程低くない限り、基本的に全てのノードが受け取ることになります。これが「フラッディング」と呼ばれる所以です。

2. メッセージキャッシュ

先ほどの図で、中央のノードが中継した TTL=15 のメッセージは、左上でも受信できるんじゃないの？と思った方、正解です。そういうループを避けるため、自分が転送したメッセージを全てのノードはキャッシュします。

メッセージ全体をキャッシュするのはさすがに効率が悪いので、全てのメッセージにはシーケンス番号という連番が設定されています。このシーケンス番号と送信者のアドレスのペアをキャッシュして、既に中継したメッセージかどうかを判別します。

この TTL とメッセージキャッシュに基づいて、とにかく「既に宛先に届いたかどうかにかかわらずネットワーク全体にメッセージを行き渡らせる」のが Managed Flooding の方針です。固定した「経路」などをもたないため、ノードが居たはずなのに突然居なくなったり、あるいは増えたりしても、特に何か新しいトポロジ変更のような処理は必要ありません。基本的には、物理的に電波が冗長に届くように配置する前提で、このような割り切りになっています。その分「不必要」なメッセージが飛び交っているため、2.4GHz 帯には余り優しく無いとも言えますが、BLE 自体の通信時間は極めて短いため、それほどでもないようです。

Bluetooth mesh のセキュリティ

いまどき新しく開発されるプロトコルなので当然ながら暗号学的に正しく設計されています。

Bluetooth mesh は 3 つの鍵で保護されています。そのうち一つはノードがネットワークに参加するときの確認に使われる「デバイスキー」で、残りの二つが実際にネットワーク上を流れるメッセージの内容を保護するための「ネットワークキー」と「アプリケーションキー」です。

メッシュネットワーク上のメッセージは全て、ネットワークキーとアプリケーションキーの二つで、入れ子上に暗号化されています。ネットワークキーは、そのアドレス空間を構成するネットワーク全体で共通のものです。ネットワークキーを知らなければ転送元、転送先を含む全ての情報を知ることができません。Managed flooding の転送処理はこのネットワークキーを復号した状態で行われ、再び暗号化されて無線送信されます。

名前の通りですが、その中のアプリケーションデータを守るのがアプリケーションキーです。この二重構造により、単にリレーだけするようなノードはアプリケーションデータを知ることができません。見通しで 100 メートルぐらい届くことがあるとはいえ、建物の内部などでは数十メートルがいいところですから、アプリケーションとして実際に役割を果たすノードの他に、電球などにリレー専用のノードをばらまくことが考えられます。そのような「メッシュネットワークインフラ」を想定しているのだと思います。

ちなみに、この二重暗号化のモデルどこかで見たことありますね、そうです、本書の前の方で特集している SORACOM も使っている LoRaWAN です。あちらも、ネットワーク全体のキーとアプリケーション毎のキーが分かれています。そもそも長距離でスター型の LPWA と、短距離でメッシュ型の Bluetooth mesh で似たようなアーキテクチャになってくるのは興味深いところです。

で、なにができるの？

注意深くプレスリリースや概要資料等では隠されていますが、そもそも Bluetooth 4.0 でのアドバタイジングは最大で 31 バイトしか入りません。というわけで、Bluetooth 4.x を使っている限りは実際の転送能力もお察しの通りです。ただし、規制の厳しいサブギガ帯を使う LPWA などと異なり送信間隔などの制限はない（かわりに混信しやすい）ので、転送能力の総量は LPWA より多くなります。

あくまで Bluetooth mesh の規格としては、途中の Upper Transport 層での暗号化時に 380 バイトという制限がありますが、それは Bluetooth 5 以降でのアドバタイジングの長さ拡張が前提となります。Bluetooth 4.x の場合は、パラメータ設定によって異なりますが、だいたい実際のデータ本体としては 1 回に 10 バイトぐらいが限界のようです。LPWA と同じく「1 ビットは血の一滴」を要求される世界ですね。

実際のメッセージで送ることができるデータは、「Mesh Model」という仕様において「モデル」という枠組みで定義されています。色々と、ありがちなデータのパターンがモデルとして定義されているので、眺めてみると「奴等はこんな利用ケースを想定しているのか」ということが大変よく分かります。というか照明制御好き過ぎでしょ。

たかが 1 回 10 バイトでも制約が少なく自由に飛ばせるので、デバイス側で必要な有効数字に落とし込んで送ったりする分には十分ですし、通知や制御にも十分活用ができると思います。LPWA 以上に、パズルが楽しい技術ではあります [53]。

[53] そしてステマにならないように書いておくと、筆者の昼のお仕事がまさに Bluetooth のメッシュ技術を実用化している会社だったりします。一緒にパズルやってくれる仲間も募集中！

なろう！フルスタックエンジニア

少なくとも日本国内（の私の Twitter）では、いわゆる大喜利ネタとして定着した感じのある「フルスタックエンジニア」というキーワードですが、今回はあらためてこのフルスタックエンジニアについて考えてみようと思います。

フルスタックエンジニアという単語そのものは、2013 年 6 月に書かれた Publickey の記事[54]によると、海外とりわけ米国のスタートアップ企業における求人広告に登場したようです。基本的には、スタートアップ企業の立ち上げステージにおいて必要とされる、ビジネスに必要とされる以下のようなスキルを持つ人材を指します。

- Web アプリケーションの開発
- スマートフォンアプリケーションの開発
- Web システムの運用
- Web やスマートフォンアプリを含むサービス全体のアーキテクチャ設計
- 情報システム

それが日本国内のスタートアップ等でも求人広告に利用されるようになり、それを取り上げたブログ[55]などを通じて認知度が上がっていったようです。

由来

フルスタック――すなわち full-stack は、いわゆるコンピュータサイエンスにおけるスタック（stack）が積み重なっているというイメージの英単語です。おそらく由来としては、IT 業界（というより Web 業界）において、OR マッパー（等の DB アクセス）や表示テンプレート処理など一通りの機能を備えたウェブアプリケーションフレームワークを指して「フルスタック」だと表現していたものが、フルスタックエンジニアという言葉が発生する背景となったのでは無いかと思います。

---

[54] 「最近よく目にする「フルスタックエンジニア」とは何だろうか？」
http://www.publickey1.jp/blog/13/post_230.html

[55] 特にこの「35 歳定年説より怖いフルスタックエンジニアしか生残れない未来とは - paiza 開発日誌」という記事が火付け役としての役割を果たしたように思います。なお paiza は転職関連サービスの一つです。
https://paiza.hatenablog.com/entry/2014/05/13/35 歳定年説より怖いフルスタックエンジニアしか生

具体的には Ruby on Rails が、様々なライブラリを設定で組み合わせる文化に対抗して、単独で完結して規約に従い自動で設定される事を売りにするために積極的に使っていたようです。Rails の場合には、単純にサーバサイドの実装に必要なコンポーネントを備えただけでなく、クライアント JS 側に対する様々な支援機能も備えていた事が、より「フルスタック」性を高めています。もっとも、その後 Rails をはじめとするフルスタック型の重厚なフレームワークと、自分自身の機能を最低限に保ち必要に応じてプラグインや別のライブラリ等を使えば良いという軽量フレームワークの流れが来た事も記憶に新しいところです。

このフルスタックという単語が英語圏でどれくらいの温度感を持つのかは判りませんが、この「これさえあればひととおり」というイメージが、スタートアップの初期で必要とされる人材像にマッチしたため、それなりに広く使われ日本語圏にも登場しました。そもそもが Web 系スタートアップということもあり、スマートフォンアプリを含む Web サービスを立ち上げるために必要なエンジニアスキルが期待されているわけです。

その後の大喜利展開は皆さんご存知の方も多いと思いますが、主に使い潰される社畜的なイメージであったり、どこまでできたらフルスタックと名乗れるのか見たいな話だったりで、Publickey の記事から 5 年近くたっても Twitter 検索では日々フルスタックネタが投稿されているのが見られます。

フルスタックってどうよ

日本における「フルスタックエンジニア」はざっくり二種類の方向性があるかなと感じます。

一つは役割としてのフルスタックエンジニアで、初期のスタートアップに代表される人が少ない組織において、複数の技術領域を掛け持ちで担当している（担当せざるを得ない）エンジニアです。とはいえ創業間もない頃ならまだしも、本来ならば採用を強化しなければいけない段階においても人不足によってそこから脱却できないということが少なからずあるようで、フルでスタック（stuck：立ち往生）するエンジニアなどと揶揄されたりします。

もう一つはスキルとしてのフルスタックエンジニアで、フロントサイド（ブラウザ JavaScript やスマートフォンアプリ）からサーバサイドまでの開発スキルと、インフラの構築・運用スキルまでを持っていれば、ひとまず当初の意味としてのフルスタックであるとされています。実務上の必要性から、小企業の情報システムに関するスキルあたりまで求められる事が多いでしょう。これは、昔から言われる「T 型人材」のパワーアップ版とも言えますし、一方で手をきっちり動かせるシステムアーキテクトの領分でもあります。

この二つは明確に分ける事に意味は無く、フルスタックのスキルを持つ人（あるいは持たない人）が、必要に応じてフルスタックの役割を任されているわけです。

やっぱりフルスタックって格好良いですよね、名乗ってみたいですよね、じゃあフルスタックエンジニアになるには、あるいはフルスタックエンジニアと名乗るにはどうすれば良いか考えましょう。

大喜利フルスタックエンジニア

ところで、Twitter などで冗談で取り上げられるフルスタックエンジニアには、以下のようなスキルが求められています。

- ハードウェア
- コンピュータサイエンスのすべての分野に精通 [56]
- 量子コンピュータ
- デザイン、作曲・作詞
- 電磁気学、数学
- 営業、顧客対応
- 地鎮祭
- イベント運用、音響操作
- 筋肉（ラックマウント能力）
- 料理、焼きそば
- 会社経営、総務、財務
- 法律、政治
- 農業

ここまでいくと空想上の TOKIO じゃねーかという感じですが、裏を返せば、これらを必要とする場面があったり、中途で取りたい凄腕エンジニアに期待したい事であったりするわけです。もっとも「それ神社に任せろよ」という地鎮祭みたいなものも複数人から上がっていて、なるほどと思わされます。苦労したんでしょうかね。ざっくり分類すると、以下の 3 つぐらいに分ける事ができそうです。

- コンピュータサイエンスの範疇
- 一般的なコンピュータサイエンスの範疇を超えるが、IT 企業でエンジニアが知っていると相乗効果が得られるもの
- そもそもエンジニアに担当させる理由が薄いもの

3 つめは完全にネタというか、実際にそういうものを求めた組織があったり無かったりするんでしょうが、それはさておき 2 番目の領域に含まれそうな、たとえばデザインであったり、法律や政治であったり、営業・顧客対応、会社経営といった領域は、何らかの形で IT 技術と組み合わせて価値が高まる場面が出てきます。なお、いわゆる業務知識と呼ばれる部分は人それぞれなので、この議論からはひとまず外します。

[56] 元ネタは Preferred Networks 社の採用募集

フルスタックエンジニアを目指すに当たっては、この 1 番目と 2 番目をバランス良く、世の中の需要に合わせてスキルを揃えていくとよさそうです。

必要なスキル（コンピュータサイエンス領域）

一般的な Web サービスであれば、まず中心となるのがウェブアプリケーションの開発能力、スマートフォンアプリの開発能力、フロント JS の開発能力、サーバインフラの構築・運用能力の 4 本柱が基本となるでしょう。サービスによっては、フロント JS とスマートフォンアプリは必要が無い事もありますが、そのような場合はそもそも「フルスタックエンジニア」とあまり呼ばないため、少なくともどちらかはできる必要がありそうです。

スマートフォンアプリについては、Android と iOS でシェアのほとんどを占めているので、その二つができれば良いわけですが、ウェブアプリとしてスマートフォンアプリを実装できる Apache Cordova だとか、iOS・Android どちらも.NET 技術で開発できる Xararin のようなものも出てきているため、それらを覚えておくとひとまずフルスタックに早く近づけそうです。

サーバインフラの運用も、クラウドネイティブな 2018 年ですのでもちろんパブリッククラウド上で安定運用ができれば問題ないでしょう。

さらなる広い領域を押さえた格好良いフルスタックエンジニアを目指すのであれば、ハードウェアや、ソフトウェアの低レイヤーの部分を狙っていきましょう。今はまさに IoT ビジネス立ち上がりの時期であり、それらがわかるエンジニアの人気はどんどん上がっています [57]。

新しい技術領域もまたどんどん登場しています。ガートナーのハイプ・サイクル [58]を眺めると、機械学習系の領域はちょうど過剰な期待のピーク紀を迎え、実際の事例に収束する幻滅紀を迎えています。これから来るものというと、筆者自身も推しているサーバーレスであったり、量子コンピュータ、会話型ユーザインターフェース（VUI）あたりは押さえておいて損が無い領域と思います。

必要なスキル（コンピュータサイエンス外）

コンピュータサイエンスと直接関わりが無くても、ビジネスをする企業におけるエンジニアとして、おさえていると「つぶしがきく」領域があります。

---

[57] なにより筆者の会社でも欲しいです。

[58] 「ガートナー、「先進テクノロジのハイプ・サイクル：2017 年」を発表」
http://www.gartner.co.jp/press/html/pr20170823-01.html

間違いが無いのが会社の運営に必要な部分で、例えば経理／財務の話は自分で持つサーバやネットワーク機器の扱いにも関連していきます。簿記資格を取って仕訳を行う必要は無くとも、基本的な減価償却の考え方などは情報システムの業務知識の一つとして知っているべきです。

デザインに関しても、もちろん餅は餅屋ですから実際にデザインをするのはそのセンスを持つデザイナーが行うべきです。ただそのデザインをアプリケーション上で実装する必要がありますし、逆にアプリケーション側の制約によってデザイン側を修正する必要があれば、デザイナーとデザインの世界観で話せればスムーズに議論が進みます。

要するに、「自分がそれを担当できるほど詳しく必要は無いが、その領域の世界観を理解し、その世界の言葉で会話ができること」が重要です。

はったり駆動フルスタックエンジニアのススメ

よく考えると、この「自分がそれを担当できるほど詳しい必要が無いが、その領域に対して一定の理解がある」ということは、コンピュータサイエンスの領域でも持ち込む事ができます。

人生の時間は有限なので、全ての技術領域で十分に活躍できるほどスキルを引き上げられる人はそれほど多くありません。ですが、数多くの技術領域に対して、「その分野で今できる事は何か、それにどれくらい掛かるのか大変なのか、何は難しいのか、あるいは現状では実現できないのか、より深く知るためにはどうすれば良いか」を浅く広く知る事はできます。

IT の発展によって、今は「学習の高速道路」が敷かれたと言われます。もちろんその先での大渋滞として一流になれるほどでは無いエンジニアの大量発生も取り沙汰されますが、フルスタックエンジニアを目指す上で全ての領域で一流である必要は薄く、必要になってから高速道路に乗るので十分です。

そこで、普段業務で使っていない技術領域についても個人的に「遊んで」おいて、高速道路の入口まではたどり着いておきます。そうすると、その技術で「何がどこまでどれぐらいでできるのか」という勘が働いてきます。そうしたらしめたもので、その技術をフルスタックエンジニアのポートフォリオに入れてしまいましょう。実は世に言うフルスタックエンジニアの大半は、普段から遊んだ事のある技術の幅の広さによって「はったり」を言っておいて、実際に業務で必要になってから高速道路でつじつまを合わせる「はったり駆動」であることが多いのではないかと考えています。

つい先日、「知識の壁」「行動の壁」「気づきの壁」「技術の壁」「習慣の壁」という段階を明確に示した図が SNS[59]で紹介されていましたが、はったりをきかすために普段から「遊んで」いれば最初の 2 つか

[59] https://twitter.com/inuicpa/status/983638176534163456

3 つの壁を既に乗り越えているわけで、業務として必要になってから「技術の壁」は超えれば良いというわけです。

最初の知識の壁を越えるために、様々な領域の勉強会などに参加してみるのも良いです。そもそも知らない事に対しては努力を始められないし、はったりも言えませんからね。

フルスタックでも基礎がだいじ

一流の一歩手前まで学習の高速道路が敷かれていると言っても、そこを最高速度で走るために必要なものがあります。それはコンピュータサイエンスの基礎知識です。あるとないとでは、原チャリとスポーツカーぐらい違います。

基礎知識といっても、別にプログラミング言語の数学的構造や、シリコンの物性を知る必要まではありません。それらは「そういう分野がある」ということまで知っていれば良いですし、それらにもまた高速道路があります。

その一方で、ある程度中身に踏み込んで理解しておくべき分野があります。

- 情報理論、データ構造とアルゴリズム
- プログラミング言語における、関数型や型付け有無など最近のトレンド、向き不向き
- 分散コンピューティング、並列コンピューティング
- IP をベースにしたコンピュータネットワークの仕組みと、現在の発展方向
- 暗号理論
- OS の仮想化技術、コンテナ化技術
- ID と権限の管理、連携技術

これらの分野の知識は、あらゆる分野の「高速道路」においても倍掛けで効いてきます。これら基礎知識の領域にも高速道路は敷かれているわけで、是非追いついておくのがお買い得です。

フルスタックエンジニアを目指そう

こうやって様々な技術領域に触れ、遊んでおくと、急に新しいチャンスが来たときにそれを掴む事ができるようになります。

それは会社として相談された新しい案件かも知れないし、プロジェクト内の役割に過ぎないかも知れないし、あるいは転職のお誘いかも知れません。あるいは自らチャンスを作り出す提案を周りに仕込んでいく事もできます。

周りからできると思われている既存と同じ仕事をただ受けるだけのエンジニアと、自らできる事を（はったり交えてでも）増やし組織全体の可能性を広げられるエンジニアで、待遇はおおきく変わってき

ますし、もし変えない組織に居るのであれば今すぐ転職を検討すべきです。後者のエンジニアは常に業界全体で不足しています。

ことソフトウェアエンジニアという業界において、技術的な停滞はごく少数を除いて低待遇への高速道路です。いいですか、いまどきの Web アプリケーションのパターンを提示して大きな影響力を今なお持つ Ruby on Rails が登場して 14 年、Amazon が EC2 をリリースして 11 年、ビッグデータ処理の幕開けとなった Apache Hadoop のリリースから 6 年、HTML5 の勧告から 3 年半、今もっとも広く使われている（使われ始めている）言語である ECMAScript（JavaScript）に至っては、仕様は毎年 6 月に改訂されブラウザやサーバサイドランタイムの実装速度と競い合っているのが、現在のコンピュータサイエンスにおける速度感です。この流れに対抗していくためには、フルスタックエンジニアという生き方は、日常的に大喜利で揶揄されるほどひどいものではないと思います。

どうです、あなたもフルスタックエンジニアとして生きませんか？

htc vive 買ってみた

大昔から.hack だったり SAO だったりの影響で VR に興味は持っていたのですが、Microsoft の Windows Mixed Reality 展開であったり、昨年からのバーチャル YouTuber（VTuber）の波を受けて VR あらためて興味を持っていたのですが、遂に奮起して htc vive を買ってみました。というか、シロちゃんかわいいです。順番に殴られたい。

買ってみてからよく分かったのですが、VR にはざっくり二種類あります。

一つは「360 度動画」で、しばらく前から RICOH THETA の影響で一気に普及したあれです。いわゆるスマホ動画もほとんどがこれで、言ってしまえばパノラマ写真の発展系です。VR ゴーグルを付けた人の視線の向きに応じた画像が見られるというものです。ジェットコースターに載せられているデモを見た事がある方も多いと思います。

もう一つは「ルームスケール VR」で、視線の向きだけでなく視点（というか頭）の位置を VR 空間内に反映する事（位置トラッキング）ができるものです。360 動画ではあくまでカメラの場所は固定ですが、こちらは実際に移動する事ができるわけで、与える価値として全く別のものという事に体験してみてようやく気づきました。スマホ VR でも、iOS11 の ARKit 等で位置トラッキングへの対応が始まっているようです。ただ特殊なカメラで撮影すれば良い 360 度動画と異なり、「VR 空間」を意識したコンテンツを用意する必要があります。

今回の htc vive はルームスケール VR が可能な機種で、あらかじめ 2 箇所に設置したベースステーションから赤外線でトラッキングをしているようです。実は最近上位バージョンも出ていますが、そのおかげで値下げしたタイミングで変えたのでハッピーです。

設置

とりあえず勢いで買ってしまったので、まず部屋を片付ける必要があります。VR 空間として利用するスペースは真四角である必要は無いのですが、内側に 2m×1.5m の長方形が取れる必要があります。実はすぐに用意できる空間がぎりぎりで、かなり試行錯誤が必要でした。

まず高い場所にベースステーションを設置するのですが、私の部屋はもともと 2 つの部屋を繋げてある関係で L 字形になっています。そのため、上手く設置しないとベースステーションからコントローラに赤外線が届かない死角ができてしまいます。ひとまずはと思い、多少狭くても死角ができないようにベースステーションを設置したのですが、うまく向けたつもりでも死角がちょいちょい生まれてしまい、ベースステーションの角度を細かくいじってようやくそれなりにコントローラに届くようになりました。この状態で、コントローラを手に持って動かして VR で使う空間の広さを辿っていくのですが、かなりぎりぎ

りだったため、内側の 2m×1.5mが取れないと何度も言われて 10 回ぐらいトライしてようやく成功しました。そもそも壁際の荷物をどかさないとだめだった感じですね……。

ここまでできればあとは、HMD をかぶって VR 空間の中でチュートリアルとかをこなします。操作もとても分かりやすいです。きちんと推奨通りの性能を PC 側に確保した事もあって、身体の移動との遅れも感じられない程度で、奮発したかいがあるというものです。

VRChat

で、やってみました VRChat。

正直ゲームとかどうでも良いから VRChat 自分で試して見たかったのが今回の動機です。

ぶっちゃけ vive 買ったのが技術書典の一週間前なので、軽く VRChat 入ってみてあちこち眺めて見て回った程度なのですが、ネットゲームとかを長くやっている人は、「ゲーム内で一緒に居る人が確かにそこに存在する感じ」という表現が伝わればと思うのですが、位置トラッキングによる VRChat はこの感覚を容易に達成していました。いわゆる「臨場感」とかいう指標とは全くの別次元の「存在感」があります。時間も無いので敢えてデフォルトアバターのままふらふらしていましたが、噂通りの女の子キャラクターの聖地状態で、たまにでっかいよくわからないものが動いたりロケットが飛んでいったりしていますが、この感覚は、おそらくルームトラッキングでない VR でも、アトラクション等でお客さんとしてプレイするのでも体験できないものだと思います。

VR と別の技術の組み合わせ

いろいろ他の技術とかと組み合わせてなにかできないかなと思っていて、たとえば Alexa みたいな VUI（音声インターフェース）だったり、ドローンを VR 内と連携させて実世界側の情報を持ち込んでみたり、色々試したい事が溢れてきています。

そのためには、まず 3D 系の実装技術もおさえていかないと行けないので、高速道路はありつつもなかなかゴールは高いところにある感じですが、3D アバターを配布するためのフォーマット VRM なんてのも出てきていたりと、ちょうど新規に覚えて行くには良いタイミングのように思います。いや、本当に時期はたまたまなんですが。

Vtuber

そもそものきっかけの一つである Vtuber の波ですが、ちょうど私が vive を買ったタイミングと並行して、ドワンゴが Virtual Cast を発表してみたり、カスタムオーダーメイド 3D2 がバーチャルアバタースタジオ機能を追加してみたりと、VR 空間でキャラクターになりきるためのツール整備が一気に進みました。

時間不足でまた Virtual Cast の起動試験ぐらいしかしていないのですが、こんど身内向けで色々遊んでみようかなと思っている次第です。興味ある方は某#ssmjp の Slack に入っていると良い事あるかも？

ViveTracker

vive だと、ViveTrack というのを追加する事で関節や足だったり、全く別の物体だったりを追加でトラッキングできるようになっています。個人的には ViveTracker 追加してから本番なのかなぐらいに思う事があります（要するに、頭の位置からの推定だけでなく、両足と腰に付ける事でしゃがんだ、寝転んだが取れるようになる）が、この ViveTracker が品薄状態で、なんとなくまだしばらく掛かりそうな感じです。

この辺が揃ってきたら、できる事がどんどん増えて行きそうな感じがします……！

Microsoft MVP Global Summit 行ってきた

ありがたいことに、クラウドコンピューティング全体における活動を評価していただいて、Azure 領域であまり活動していないにもかかわらず Microsoft MVP Award をいただいた 2017 年 1 月でしたが、2018 年 3 月に開かれた Microsoft MVP 向けのイベント「Microsoft MVP Global Summit」に参加してきたのでそのレポートです。

Microsoft MVP という制度をご存知で無い方も多いと思いますが、これは Microsoft がその製品に関して「コミュニティにおいて」活動している人を表彰する制度で、一度受賞しても毎年また更新審査される継続性の高い枠組みです。毎年の年号が書かれた輪っかを刺せるごっついトロフィーとかもらえるほか、なんとなく周囲にはったりが効きます。フルスタックエンジニアにもってこいですね。

Global Summit はそんな Microsoft MVP に対して、製品の最新情報に関するセッションと、製品の担当者に直接話せる機会を作るために用意されている年 1 度の機会です。このイベントはたいへん NDA が厳しく、所属組織への報告はおろか、参加者同士でもカテゴリ等によって参加できるセッションが異なることもあり隣同士でも情報交換が禁止、というレベルです。

というわけで、残念ながら内容に関する報告はありません。

行くまで

Global Summit は、基本的に丸一週間シアトルに行きっぱなしのイベントです。

MVP の特典として、基本日程（4 泊 5 日）の宿泊費は Microsoft が負担してくれます。ただし、カテゴリによっては基本日程に加えて 2 日間セッションがあるため、自費でホテルを取る必要があります。また、交通費は自費ですし、そもそも一週間会社を休む必要もあります。

なかなか参加ハードルが低くないこのイベントですが、今回の参加にあたって往復航空券と追加日程のホテル費用を会社で出していただくことができました。前述の通り NDA によって会社に直接的なフィードバックができないにもかかわらず、希望通りの範囲を負担していただいた株式会社 WHERE

まじ良い会社です(ステマ)。成田・シアトル直行便の往復航空券だけで 10 万円以上掛かってくる[60]ので、本当に感謝しています。

会社との調整と並行して、Global Summit の登録がはじまります。タダ宿泊できるホテルは Microsoft から提示され Global Summit 申込時に確保しますが、パーティ等のメイン会場である Hyatt Regency Bellevue などの人気ホテルは取り合いになるため、必然的に Global Summit への申し込み自体もなのは完売みたいな熾烈な争奪戦でした。どこまで書いて良いか判らないのでボカしますが、とにかく、なのは完売みたいな感じでした。

私は運と粘りによって Hyatt を確保できたので、あとはシアトルに行きます。

行き

実は公式ツアーがあったり無かったりするのですが、延泊組は帰りの飛行機の都合で使えないため自ら取った飛行機に乗る事になります。が、直行便(の ANA)はそもそも一日一往復しかないので、搭乗ゲートにたどり着くとどこかで見たような人達をぞろぞろ見かける事になります。正直、米国旅行に不安しか無かったので本当に心強かったです。

あと、A787 の窓が本当に何一つ GPS を絶対通さないマンで、せっかくの長距離移動の GPS ログ[61]が取れなかった悲しみと悔しさをここに記しておきます。

シアトル・タコマ国際空港に着くと、慣れてる人がささっと Lyft(Uber のもっとイケてるやつ)でささっと車を手配してくれました。空港の駐車場に、Uber と Lyft 向けの一時駐車エリアがはっきり名指しで用意されていたのが興味深いです。個人タクシーのような感じで完全に社会に受け入れられ定着しているのを実感します。

[60] 日程発表からすぐに割引チケットを確保するともう少し安かった模様。パックツアーとかにうまく組み込めるともう少し安くなるのかも知れないですが、海外旅行初心者なのでそういうのはチョットムズカシイです。

[61] 「世界の霧」というスマホアプリで、過去に行った事のある場所を記録しています。

Global Summit 参加

セッションの内容は書けないので、まあ雑多な事を。

おやつとかドリンクとかでますが、Cucumber Watch（きゅうり水）という日本だとそれカッパかよみたいな水がしれっとありましたが、さほど不味くもなくふつうでした。というか色んなフレーバーウォーターがあったのですが、死ぬほど不味いのが一つだけありましたね。

シアトル・ベルビュー滞在

基本的に宿泊はシアトル郊外のベルビューというエリアなので、ベルビューを起点にマイクロソフト（あのレドモンド）までバスでドナドナされたり、世界最長の浮き橋を渡ってシアトルまでバスでドナドナされたりします。いまちょうどシアトルからベルビュー経由でレドモンドの方に向かう鉄道を建設しているようで、あと何年かするともっと楽に行き来ができるようになるようです。といっても、バスもプリペイドカードで支払い可能で、長距離バスはシアトル市街地では地下のトンネルの中を走っていくのでさほど不便でも無いです。

ベルビューにはラーメン屋も何軒かあり、山頭火と輝月があります。おすすめは輝月。

あとは、シアトルに Amazon GO ができていたので行ってきました。いわゆるコグニティブサービスを活用した無人コンビニで、自動改札みたいなゲートにスマホアプリが表示した QR コードをかざして、あとは監視カメラ等の映像で棚から持っていった商品を識別、ゲート退場時に決済を自動で行うというものです。商品はすぐ食べられるものも多く、すぐ横にイートインスペースがありその場で食べる事ができます。なんの変哲も無いゲートを商品持って出ただけで、すぐ横で買い食いしてしまうの、無駄に罪悪感が半端ないです。いや、Microsoft のイベントに行って Amazon で喜んでどうするんだって話ですが、周りの MVP みんなこぞって Amazon GO してました。
1 月にできたばかりだし仕方ないね。

あと、サンケイビルを見かけました→

Microsoft 公式ショップ

Microsoft のキャンパス内に公式ショップがあります。

ちょうどすぐ前から Minecraft にはまっていた事もあって、マイクラグッズを右から左に買っていた気がします。TNT の四角いマグカップは、ねこの水コップになっています。（追記：残念ながらだんだん底が加水分解してきてしまったので、お取り上げになりました。）

帰り

帰りはちょっと慣れてきた事もあって、GPS ログ目当てに一人で早めに出てシアトルから鉄道経由で空港まで移動しました。シアトルでは、鉄道と路面電車とバスが都市部で同じトンネルを走っているのが本当に興味深かったです。

まあとにかく言える事は、ANA やっぱり楽というか安心ですね……。

まとめ

というわけで、Global Summit（に付随したシアトル旅行）は楽しかったです（小並感）。

2015 年振り返りと 2016 年予想

今年個人的に心に残ったテーマと、来年盛り上がりテーマをつらつら書きます。

Superfish/eDellRoot

PC ベンダーが、独自ツールのために脆弱な独自ルート証明書を OS に導入していた事件です。サイトでも、「Superfish/eDellRoot が危険な理由」[62]として取り上げました。

擁護のしようも無く、ひどい事件でした。しかも 2 月の Lenovo（Superfish）で勘弁してくれよと思っていたら 11 月には DELL からも同様の問題が見つかるというまさかの二段落ちです。

どちらも独自ルート証明書に共通の秘密鍵を付けてしまったために脆弱になってしまったのですが、細かいところで違いがありました。Lenovo の Superfish はそもそも HTTPS サイトにも広告を差し込むために MITM（Man-in-the-middle）攻撃を仕掛けるのが目的でした。したがって、各 PC に秘密鍵が置かれていることが必要になります。こういったアドウェアを仕掛けることの是非はさておき、これを安全に実装するのであれば、PC ごとに秘密鍵を変えてルート証明書を生成する必要がありました。

一方で DELL の eDellRoot（および同時に発覚した DSDTestProvider）はサポート用のツールに含まれていたもので、細かい利用目的は明かされていないようですが、おそらくはオンラインサポート時にサポート用システムに接続する時に何らかの理由で独自ルート証明書を使っていたのでは無いかと考えられます。そのためルート証明書を導入したが、その時に秘密鍵ごとインポートしてしまったのでは無いかという想像です。こちらは、単純に証明書だけをインポートすれば安全だったと言えます。

どちらも、あまりにもカジュアルにルート証明書を弄ったために、痛いしっぺ返しを食らうことになりました。

うるう秒

残念ながら当面の存続が決定してしまった「うるう秒」ですが、今年の 7 月 1 日に挿入されました。サイトでも「うるう秒まとめ [63]」「Linux のうるう秒おさらい [64]」として取り上げました。

---

[62] http://d.hatena.ne.jp/nekoruri/20150220/superfish

[63] http://d.hatena.ne.jp/nekoruri/20150521/leapsecond

[64] http://d.hatena.ne.jp/nekoruri/20150629/leapsecond2

また、一緒に検証をした某氏が某技術ブログで取り組み内容を書いています[65]。

ほとんどの環境では 59 秒が二周繰り返されるだけでそれほど気にしなくても良かったはずなのですが、前回 2012 年 7 月のうるう秒で Linux カーネルの不具合でサービス停止にも繋がる大きな問題があったため、今回は真面目な検証と対応を余儀なくされました。

うるう秒は黙っていると「Leap Indicator」として入り込んできます。うるう秒による不具合を完全に防ぐためには、かなりの努力が必要でした。その苦労を「非うるう秒三原則」としてまとめています。下位の ntpd と同期させたまま、ゆっくり時間をかけて 1 秒を受け入れるスクリプトも技術ブログの方で紹介されているので、万が一次回のうるう秒があった時には参考にしてください。

あと、CentOS 4 はそろそろなくしましょう（真顔）

通信の最適化

だいぶ前から少しずつ声が大きくなり、ついに今年の夏に爆発した「通信の最適化」の議論がありました。携帯電話事業者が、帯域幅を絞るために通信内容に含まれる画像や動画の再圧縮を行っていたというものです。

この問題は、直感的にはアウトっぽく見えるけど、法律的にはそこまで単純では無いという興味深いテーマです。大きく二つの論点があります。ひとつめは、大前提として、そもそも「通信の秘密」は侵害しているがその違法性を別の理由によって阻却しているという建前があり、この建前が本当に正当なものかという話です。もう一つは、「画像や動画の再圧縮」によってアプリケーションで問題が出ている、という現実的かつ実装上の問題です。

他にも、数多くの論点が存在して自分が気にしている論点を主張しあうのでカオスだったという経緯で、サイトでも「「通信の最適化」の論点[66]」として取り上げました。

TLS デフォルト時代

もはや TLS 無しでは機能しないインターネットに大きく二つの変化がありました。

一つめは、正確には昨年末の話ですが SSL 3.0 プロトコルそのものに脆弱性が見つかり、「SSL」と呼ばれるトランスポート暗号化が全て「オワコン」になったことです。脆弱性対応のためにはサーバ側で SSL 3.0 自体の対応を切る必要があったため、この一年でだいぶ SSL 3.0 の非対応化が進みまし

[65] http://ameblo.jp/principia-ca/entry-12046176524.html

[66] http://d.hatena.ne.jp/nekoruri/20150715/transcoding

た。2015 年 12 月 4 日時点の SSL Pulse によれば、昨年 10 月には 100%近かったのが 30%前後まで落ちたようです。

SSL 3.0 もオワコンしたので、そろそろ「いわゆる SSL」とよばれるこの TCP 暗号化トランスポートプロトコルのことは TLS と呼ぶべきでしょう。TLS で暗号化されたウェブサイトのことは HTTPS サイトで良いと思います。

もう一つは、常時 HTTPS 化です。2014 年 8 月に Google が「HTTPS をランキング シグナルに使用します」というアナウンスを出してから流れが始まり、Apple がさらに攻めた「App Transport Security（ATS）」を打ち出して業界全体で話題になりました。

ATS では HTTPS 化だけでなく、Perfect Forward Secrecy（PFS）という性質を持つ暗号化アルゴリズムの利用を強制します。これは SSL 証明書の秘密鍵が漏えいした場合でも、それだけではあらかじめ保存しておいた過去の通信を復号することはできないという性質です（秘密鍵を取得してから MITM することは可能）。昔は、まさか「とりあえずトラフィック全保存しておこう」なんて攻撃者は現実的には居ないだろうと思われていたのですが、NSA 情報漏洩事件で実際に行っている可能性が指摘されてからは PFS が一気に注目されるようになりました。

もはやあまり DNS も通信路も信頼できる牧歌的な時代では無いので、HTTPS デフォルト化は時代の流れですね。HTTP/2 でもユーザ側の通信は原則 TLS ですし。

日本年金機構

情報セキュリティで 2015 年最大の事件が、5 月に発生した日本年金機構からの情報漏洩であることに異存がある人は少ないでしょう。

既に調査結果報告も出ています[67]が、とにかく「現場担当者のリテラシー」などという信頼できないものに頼らず、システムとしての情報セキュリティが重要であるという認識が広まればと思います。

セキュリティ人材の育成

手前味噌ながら、セキュリティ・キャンプ全国大会にて「クラウドセキュリティ基礎」「クラウドセキュリティ演習」として、サービス運用者から見たクラウド環境でのサービス運用とセキュリティ施策の話をする機会を頂きました。

[67] http://www.nenkin.go.jp/info/index.html

いろいろな大人の事情(察せ)で、事前課題がさくらのクラウド、当日の演習が AWS になってしまったのは大変後悔しているところです。きちんと同じ環境で課題を出せていれば、当日はもっと深いところまで遊べると思うので、もし次回また話す機会を頂けたらもっともっと練り込んでクラウドとセキュリティの運用の楽しい話をしたいです。

また、まさか 2015 年にもなってインターネット回線が落ちるとは思わなかったので、AWS に接続できない場合フォールバック策を作り込んでいなかったのも痛かったです。実のところ、配布 PC 上には Vagrant や基本的な Docker のイメージ群など直前で内容を差し替えてオフラインでもなにかできるくらいには仕込みはしていたのですが、完全に準備不足でシナリオ作りまで手が回りませんでした。ごめんなさい。

Docker と HashiCorp

もう今年の動きは zembutsu さんのスライド[68]を上から見てくださいという感じのインフラ DevOps 界隈です。

丁度 1 年前に「あと半年もすれば仮想ネットワークベースのコンテナ間接続手法が出揃う気がするから、特に Docker に愛のある人で無ければ、それまで単に 1 ホスト 1 コンテナなただのデプロイ手法として既存の管理方法をベースにノウハウ貯めるか、PaaS 経由で使うぐらいでちょうど良さそう」という予想を立てていたのですが、だいたいその通りになった感じがします。

Docker はその構成上「ホスト」に捕らわれる場面が多かったのですが、オーバレイネットワーク等でのネットワーク仮想化も現実感を増してきました。個人的にはあまり魅力を感じませんが、Docker コンテナをベースにしたサーバーレスアーキテクチャみたいな選択肢が来そうな気がしています。

フォグコンピューティング・エッジコンピューティング

ここからは来年の注目キーワードです。

今年はものづくり系に楽しいおもちゃが振ってきた一年でした。安価な契約と M2M 向けの付加価値を提供する SORACOM やさくらの IoT プラットフォーム、5 ドルコンピュータ Rasberry Pi Zero、1 万円を切った Windows 10 搭載スティック PC、とにかく「小っちゃいって事は便利だねっ」を体現するデバイスが立て続けに登場しました。

この数年はクラウドに以下に集約して最適化を図るかという流れが大きかったですが、小さくてもそれなりの計算機資源と接続性を持つコンピュータが揃ってきたので、来年あたりからはクラウド上の計

[68] http://www.slideshare.net/zembutsu/presentations

算機資源をさらに活かすために、こちら側にもコンピュータをばらまくというモデルが広がるのでは無いかと予想しています。

こういったモデルを、Cisco は雲より近い所にあるフォグコンピューティング、NTT はネットワークの端っこにあるエッジコンピューティングと呼んでいるそうです。

マイナンバーと情報セキュリティマネジメント

マイナンバー、ついに 1 月から利用が始まります。

個人的には、マイナンバーにはまず政府内での名寄せの実現に注力してもらい、マイナンバーカードの利活用の話はもうすこし冷静にまともな議論を経てから実施して欲しいところです。

制度設計やシステム設計にいろいろ議論が尽きないマイナンバーですが、ひとまず動き出してしまえばかなり厳しく法整備がされているので、マイナンバー管理に合わせて情報セキュリティマネジメントへの理解が進むのではないか、というより進んで欲しいです。

また、物議を醸した IPA の「情報セキュリティマネジメント試験」も来年春から実施されます。余裕があったら挑戦してみようかなと思っているところです。

Back to the Future における「未来」

今年は、皆さん大好き（ですよね？）な Back to the Future における「未来」の年でした。スピルバーグの息子はジョーズ 19 を作っていませんが、ジョージ・ルーカスはまだ現役でスターウォーズの新作作って 3D で公開した、そんな 2015 年でした。

1985 年に想像した 2015 年は、車が空を飛ばない以外は案外あたっていたのでは無いかなと感じます。今から 30 年後の 2045 年は一体どういう社会になっているでしょうか。

2016 年振り返りと 2017 年予想

今年個人的に心に残ったテーマと、来年盛り上がりそうなテーマをつらつら書きます。

「サーバーレスアーキテクチャ」の躍進

クラウドコンピューティング業界で 2016 年もっともバズったキーワードは「サーバーレス」でしょう。この「めもおきば TechReport」でも継続的に取り上げており、ありがたいこと「サーバーレスの薄い本」は国内のサーバーレス関連イベントでも頒布やダイジェストの講演などの機会を頂きました。

「サーバーレス」が何を指すかは前巻に譲りますが、今年は道具が揃ってきたことで様々な「先進的な事例の紹介」が紹介されるようになってきました。とはいえ、まだまだ「先進的」でありコモディティ化までは道半ばといったところです。

第一に、サーバーレスアーキテクチャでは旧来のシステム以上に様々なクラウド上のコンポーネントを最大限に有効活用します。今までのソフトウェアの構成管理という枠組みから一歩踏み込んで、様々なクラウドコンポーネントの構成や設定などを「ソフトウェアの一部」として開発プロセスに組み込む必要があります。これまでに登場した Terraform や、CloudFormation、Azure リソーステンプレートなどは、DSL を元にクラウドコンポーネントを構成・設定するという部分は実現しますが、それらを実際に開発プロセスの一環として、CI で様々なテストに組み込んで回したり、ローカルかそれに近い環境での自由な試行錯誤をしたりということを考えると、まだまだ開発支援の余地があると言わざるを得ません。今はあくまで FaaS が中心となり、そこにコンポーネントの定義を Terraform や CloudFormation で組み込んで一括管理するという枠組みがほとんどですが、アーキテクチャとしてはコンポーネントを繋ぐ FaaS という形態であるべきで、一つのシステムを DSL で定義し、そこに含まれる複数の FaaS を一括管理する、という形態の構成管理が 2017 年で登場するのではと予想しています。

第二にフルマネージドなコンポーネント、特にデータストアの「癖」が強く、利用ノウハウが貯まっていないことがあげられます。サーバ単位での管理を廃して自由にスケールするシステムを実現するためには、様々な制約(「癖」)を受け入れる必要があります。たとえば AWS であれば Lambda と最も良く組み合わされるデータストアは、やはりサーバという単位を廃した DynamoDB ですが、これは KVS であり、プライマリキーの設計方法に強い制約を持ちます。基本的には限定された操作でなければ実用できる性能が出ません。あるいは AWS に限らずみんなが大好きなデータストアに Google BigQuery がありますが、こちらは力業でそこそこ何でもできてしまう一方で、きちんと設計せずに安易にクエリーを投げると、力業が容易に効いてしまう分だけ課金額も容易に跳ね上がってしまいます。データストアに限らず FaaS の実行環境としても、例えば AWS Lambda であればメモリを使わないからといって 128MB に設定しておくと予想以上に性能が出ず、多めに割り当てた方がかえって処理が早く終わり消

費量が削減できるという場合もあります。このような、スケーラブルなフルマネージドコンポーネントの特徴や、それを活かすためのベストプラクティスが今後 1～2 年で出揃っていくでしょう。

この二つの課題に対する進歩に従って、今後のサーバーレスアーキテクチャの利用が進んでいくものと思われます。

コグニティブと Bot

今年注目されたキーワードの中でもとりわけバズワード感が高いのが「コグニティブ」では無いでしょうか。コグニティブという単語自体は「認知」と訳される単語で、元は IBM あたりが広めて最近は Microsoft や Amazon がそれに乗っかったという形です。そんなはっきりとした共通定義に欠ける状況ですが、コグニティブコンピューティングと言ったとき、大きく二つの特徴があるものを指すようです。

一つは「人間がやるように入出力をすること」です。これまでは「人間が機械に合わせる」すなわち、マウスポインタやキーボード、タッチパネルなどを用いて機械が提供するユーザーインターフェースに人間が応えるというインターフェースでした。コグニティブコンピューティングでは、人間の音声や、自然言語文、画像などを、今度は機械の方が頑張って理解しようとするわけです。いわば syntax（文法）を人間に寄せるわけです。

もう一つが「人間の考えるように入出力をすること」です。人間は膨大な情報を脳が経験として貯めることで、様々な推論を行ったりします。これからは機械も同じように大量のビッグデータを元に推論を行うことで、人間のやり方のまま手助けができるようになります。こちらは、いわば semantic（意味論）を人間に寄せていると言えるのかも知れません。

この一年は、親戚の Zo さんが生まれたらしい「女子高生 AI りんな」の LINE アカウント密かに人気でしたが、とにかく動きや考え方が「人間っぽく見える」、そしてそういう形で人間を手助けしてくれる（りんなちゃんが手助けをしてくれるかは知りません）という世の中が来るのでしょう。

コグニティブコンピューティングと会わせて、チャットツールを利用した「Bot」の活用が増えてきました。選択肢と入力欄の機械っぽいインターフェースと、自然言語で自由度が高いが誤解される可能性を常に孕む人間っぽいインターフェースで、どちらが人間の活動を支援するのに相応しいのかは気になるところです。

IoT とセキュリティ

電力自由化と共に全国的にスマートメーターの本格利用が始まった 2016 年ですが、様々な形態の「IoT」においてセキュリティ上の課題も明らかになってきています。これまでは物理セキュリティによって閉鎖性が確保されていたため、性能が低い組み込み機器ではソフトウェア設計として継続的なセキュリ

ティ対策という視点が後回しにされていました。それらをいざネットワークで接続して全体として一体のシステムに昇華しようとしたときに、その欠如が開発プロセスや開発者の設計スキルなど様々な面で追いついていないのが根本的な原因です。

一番問題となっているのは、今までは「物理セキュリティを突破できる管理者」しか利用しなかったため、利用者や動作主体を識別するという概念そのものがないことでしょう。一つのデバイス内の動作は一つの動作主体でも構いませんが、ネットワークを介して別のなにかとコミュニケーションをするときには、相手に自分を信頼してもらうために「自分は何なのか」を適切に示す必要がありますし、「何なのか」を客観的に識別するための全体の枠組みが必要です。つまりは正しい ID 基板を構築して、そこに組み込み機器を紐付けていく必要があるわけです。幸いにして、暗号学的な技術要素としては結論が出ていて、可能ならば耐タンパー性のあるスマートカード等を利用して、基本的には公開鍵暗号ベースの電子証明書、適切な運用ができる場合は識別用の共有鍵（例えば 3G/4G の SIM など）を組み合わせることで実現できます。また暗号化アルゴリズムの計算処理についても、様々な組み込み機器向けプロセッサは最低限必要なアルゴリズムの処理をオフロードする機構を搭載するようになってきています。このように、きちんとシステム間や利用者を識別・認証するための状況は整いつつあります。

もう一つの大きな課題は、IoT とビッグデータを組み合わせたケースにおいて、ビッグデータ分析の結果を操作・悪用するような攻撃手法に対する守り方が確立していないことです。仮に、たくさんの気温センサーからのデータを元に空調を操作するようなシステムを考えると、特定の場所の温度センサーを「騙す」ことで、全体として空調の設定を狂わせ、人や機器の調子をおかしくする、という攻撃を想像したときに、空調の異常からセンサーへの具体的な攻撃を特定することが難しいかも知れません（例が分かりづらくてごめんなさい）。しばしば「攻撃は芸術であり防備は技術である」と言われますが、芸術のような攻撃をいかに技術体系で守り切るかという点において、IoT とビッグデータの組み合わせはかなり難しい戦場となることでしょう。

新たなる「ラストワンマイル」競争

ラストワンマイルといえば、インターネット接続の普及において電話局舎から家庭や企業までの最後の加入者線をどうするかという競争を示すキーワードでした。結果として日本ではフレッツに代表される光分岐方式によって、ほとんどのエリアにて家庭まで光ファイバを引き込める状況を実現し、ラストワンマイル競争は終息しました。これに対して、IoT という文脈でふたたび「ラストワンマイル」という競争が始まりました。

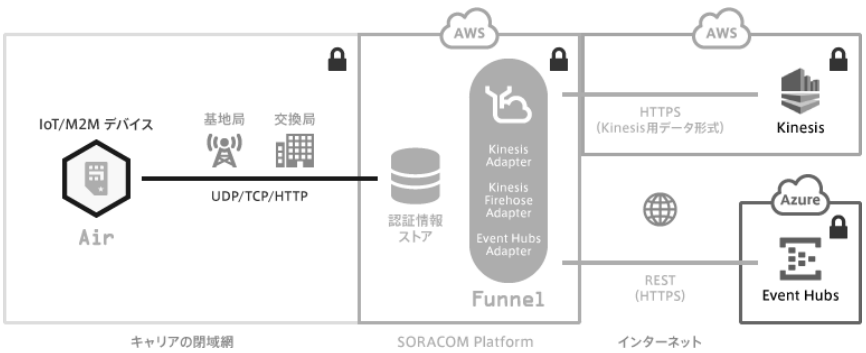
今回のラストワンマイル競争は、家庭という単位ではなくもう少し広いエリアを対象とし、移動したりばらまかれたりする IoT デバイスに対して、どのように安価かつ省電力でネットワーク接続性を提供するかという部分が戦場となっています。キーワードとしては「LPWA（Low Power, Wide Area）という分類がされます。大きく分けて、いわゆる LTE 規格の省電力仕様である Cat.M1 および NB-IoT と、

免許が不要な特定省電力無線を利用した LoRa や SIGFOX などに分けられます。国内 3 大キャリアも、NB-IoT に軸足を置きながらも、複数の IoT 向け無線接続技術の実証実験を並行して開始しています。各社の検討中の規格をまとめると以下のようになります。

種別	事業者	採用・検討中の新規格
MNO	NTT ドコモ	NB-IoT、LoRa
	KDDI	SIGFOX、NB-IoT
	ソフトバンク	LoRa、NB-IoT
MVNO	SORACOM	(LTE NTT ドコモ)、(LTE KDDI)、LoRa
	さくらインターネット	LTE Cat.1、LoRA

このように新しい通信技術の試行錯誤が進む一方で、既存技術を有効活用して IoT 向けに「接続網を再定義」する通信事業者として、SORACOM やさくらの IoT プラットフォームなどが登場しています。どちらも単なる IP ネットワークへの接続サービスを提供することを目的としておらず(実際さくらの IoT プラットフォームは、単純な IP 接続サービスとしては利用できない)、IoT デバイスからの情報をクラウドに送信したり、クラウドから IoT デバイスにメッセージを送ったり、あるいは回線接続そのものをコントロールしたりなど高水準な API を提供しています。

これは一種の IoT 向け SD-WAN(Software-defined WAN)と捉えることが可能で、IoT デバイスにとっての接続網に必要な機能を整理した結果、たとえば SIM で識別された接続元を元に認証処理屋接続先の管理を接続網がオフロードする SORACOM Beam や、より高度にパブリッククラウドのビッグデータ向けストリームにデータを投入できる SORACOM Funnel が特徴的です。



(出典:https://soracom.jp/services/funnel/ SORACOM Funnel)

確かに SIM を挿している時点で適切に認証ができているわけですから、その認証された IoT デバイスに紐付ける形で、認証情報や送信先を接続網が管理することで、IoT デバイス側には送信先や認証情報などの「デバイス固有の情報」を持たせる必要が無くなり、いわゆる「1 を 0 にする」ことが実現して

います。IoT デバイスは大量生産して広くばらまくユースケースが多く、これは大変重要なことです。SORACOM Funnel はいいぞ。

最後に、手前味噌ながら中の人が所属している某社では Bluetooth ベースのメッシュネットワーク技術を利用した TAN（Things Area Network）としてラストワンマイルの「下半分」を支える技術を開発しています。同種の技術として、古くからある Zigbee や DUST Networks などいくつかの方式が同じ領域に存在しています。「ラストワンマイル競争」と言いつつも、実際には局舎（的な拠点）から IoT デバイスに至るまでには、ゲートウェイを挟んで複数の最適な技術を組み合わせ、さらにはゲートウェイ上でエッジコンピューティングによるデータ集約も行ってよりクラウドまで実際に上がるデータ量を削減するという方向に向かうのではないかと予想しています。

エッジコンピューティング

キーワードが出たついでにエッジコンピューティングですが、昨年末に「こちら側にコンピュータをばらまくというモデルが広まるのではないか」と予想しました。残念ながら具体的な有効事例というほどは出てこなかったのですが、Amazon が AWS Lambda を IoT デバイス上で動作させる Greengrass を大々的に今年なんとか滑り込みで打ち出しました。

これをきっかけとして、まだ 2017 年は試験的ではありつつも特徴的な事例が登場し、応用が進んでいくと予想します。

FinTech

日本政府として IoT に負けないくらい力が入っているのが FinTech 分野です。Bitcoin をきっかけとした便利な仮想通貨と、クローラー型の口座アグリゲーションサービスの普及を背景に、「まず足を引っ張らないように規制緩和」という流れが進んでいます。これを受けて、2017 年には具体的な API の整備が進み、様々なユースケースが提案されていくでしょう。日本の代表的な利用者向けクロール型口座アグリゲーションサービスである NTT ビズリンクの Agurippa が登場して、MSN マネー残高照会サービスやジャパンネット銀行の JNB アグリゲーションとしてサービスを開始したのが 2002 年頃ですから、銀行口座アクセスの IT 化には随分時間を掛かってしまったなという印象です（なお MSN も JNB もアグリゲーションサービスをすでに終了しており、FinTech ブームまで持ちませんでした……）。

国内の FinTech の動きとは独立して、IT と金融分野ではもう一つ大きな動きがありました。Apple Pay の FeliCa 採用です。「ガラパゴスだ」などと揶揄されてきた一方で、自動改札を念頭においた処理の早さ・安定さなど、NFC という単なる IC カードとの無線通信規格に留まらない高い水準を達成しており、やはり技術的に優れたものが世界を征服するべきです。現状では VISA の扱いや IDm 等を利

用した簡易認証（アーケードゲームや勤怠システム等）などとの組み合わせで課題がありますが、なにしろ利用者が多い iPhone ですので 2017 年後半の次期 iPhone までには解消することでしょう。Apple が自力で普及させようとしている Apple pay と異なり FeliCa 経済圏は既に幅広く確立しているため、今後はこの FeliCa 経済圏、特にポストペイ型の iD・QUICPay を海外から来た人にどうやって気軽に使ってもらえるかを考えて 2020 年までに準備しておく必要がありそうです。また、この FeliCa の国際展開の動きと、金融機関の FinTech ムーブメントによる API 公開が上手く噛み合ってくると、日本ではなかなか厳しい少額決済などの道も開けそうな気がします。

ユーザーインターフェースの拡張

今年の後半は Pokémon GO が話題をかっさらっていきました。以前から何度も位置情報や AR を活用したゲームなどがリリースされる度に期待感だけが膨らんでいくということが繰り返されてきましたが、Ingress という（比較的）ヘビーユーザ向けのゲームを超えて、カジュアル層で広く普及した最初のリアルタイム位置情報ゲームと言えるでしょう。AR についても、ゲームデザインの本質では無いながらも、面白さを倍増させる機能としてかなり効果を上げているようです。AR がゲームに限らず便利だということはドラゴンボールの「スカウター」の時代から広く認識されていますが、ビジネス向けの AR システムもマイクロソフトが 2017 年明けには HoloLens のリリースが迫っています。たとえば JAL が飛行機の整備手順の共有に HoloLens を使った AR を検討しているようですし、SF 的な期待に負けず伸びていって欲しい分野です。

その一方で、今年はずっと VR 元年と言われています。大御所 SONY からも PlayStation VR として大衆向けの VR システムが販売開始されましたし、当初より業界を引っ張っている Oculus VR 社が Facebook 社に買収されたことからも、そもそもインターネット上の今後の一般的なコミュニケーションインターフェースとなりうる有力候補として見られているようです。とはいえ画像・動画を URL 添付できるだけのテキストツールである Twitter や LINE、Facebook がいまだに主力だったりと、なかなか「テキストベースのメディア」を超えていくのはなかなか難しいところがあるのでしょう。

AR・VR の他にも、ユーザーインターフェースの拡張としては Apple Watch などのスマートウォッチや、Fitbit などヘルスケアに近い機器、あるいは Pokémon GO 専用の腕時計型インターフェース Pokémon GO Plus など、ウェアラブルデバイスも動きがある一年でした。スマートウォッチ大手 Pebble が Fitbit に買収されるなど業界再編も起きています。その一方で Apple Watch はこれまであまり評判が良くなかったのが、FeliCa 対応で一気に好評価を得るなど、何が原動力になるか解らないところがあります。

Docker のコモディティ化

この数年はずっとアプリケーションコンテナ界隈として動きが絶えないところではありますが、もはや単純な Linux 上のアプリケーションコンテナとしては Docker が主軸から外れることはそうそうないでしょうし、Windows 上でもやはり Docker ベースになるようです。

「Docker っぽい複数サーバのオーケストレーション」という所まで踏み込むかはさておき、古い distro も駆逐されつつあり、たとえ単一サーバ上のベタ置き実行であっても Docker コンテナとして配布するのがベストという状況になりつつあります。初期の頃はアプリケーションコンテナ内で複数プロセスをどう協調させるかといった使う側の迷走も多かったですが、Docker 本体にマルチコンテナを管理する docker-compose が統合されたことで、マルチプロセスはコンテナ毎分けてディレクトリ等の単位で共有するという結論が出ています。もはや Docker というアプリケーションコンテナ技術のコア部分はコモディティ化したといっても過言では無いでしょう。

Docker コンテナ化をすることによって、そのアプリケーションがどんなファイルを書きどんな通信をしているかをきちんと管理できるようになります。それらは本来管理者が把握していなければならなかったことであり、そういった「ゆるふわだった部分の可視化」にも繋がります。遅い遅いと言われるエンタープライズ領域においても、時間をおかずに Docker を前提としたデプロイフローやフレーム-ワークに落とし込まれていくことでしょう。2017 年はもう、サーバーレスまで突き抜けないのであれば、「Docker ファースト」で構成を作らない理由はありません。機会を見て積極的に置き換えて行きましょう。

セキュリティ教育

今年は、セキュリティ教育リスタートという感じの年でした。

春には情報セキュリティマネジメント試験が開始されましたし、後半には 2017 年春より開始される情報処理安全確保支援士の詳細が発表されると 3 年間の資格維持費用として必要とされる約 15 万円をお題に大喜利が連日連夜繰り返されると言った状況です。体制としての善し悪しについては触れませんが、何はともあれそれだけ注目を浴びていることは確かでしょう。国際的なベンダーニュートラル資格 CISSP なども、じわりじわりとセキュリティ業界以外にも認知度が上がってきています。

また、「TechReport 2015.12」でも取り上げたセキュリティ耐久競技 Hardening Project の形式を踏まえたセキュリティ演習が、この一年で広く実施されつつあるようです。毎年夏の全国大会をメインに実施しているセキュリティ・キャンプについても、今年は地方大会の開催数が増えるなど、2020 年のオリンピック・パラリンピックを最初の目標に据える形で、政府の方でも色々と動きがあるようです。セキュリティ・キャンプにかぎらず、2017 年も引き続き中の人としても協力を惜しまずセキュリティ教育に関わっていくつもりです。

おまけ

今年読んだライトノベルでは、今回のサークルカットの元ネタにも採用した「本好きの下剋上」が最高でした。投稿サイト「小説家になろう」での原作連載が丁度本編終了間近のクライマックスを迎えているところですので、C91 で本誌を入手された方は、今すぐ追いかけ始めればちょうど完結のお祭りに乗れること間違い無しです。途中までは紙や電子書籍でも出ていますので、それらでスタートダッシュするのがオススメです。

クライマックス間近繋がりでは、「六畳間の侵略者！？」もオススメです。

2017 年振り返りと 2018 年予想

今年個人的に心に残ったテーマと、来年盛り上がりそうなテーマをつらつら書きます。

「2017 年予想」の振り返り

昨年「TechReport 2016.12」での 2017 年予想をふりかえってみます。

1. 「サーバーレスアーキテクチャ」の躍進
   黎明期から流行期にシフトしつつあり、間違いなく普及が進んでいます。
   具体的なテスト手法や CI など開発に関する事例も増えてきています。「一つのシステムを DSL で定義し、そこに含まれる複数の FaaS を一括管理」という予想は、2016 年末の AWS Step Functions に続いて Azure Durable Functions や IBM Composer など様々な形で登場しました。
   フルマネージドなコンポーネントについても、Azure の DocumentDB あらため CosmosDB や、AWS の Amazon Aurora Serverless など、データストアを中心にサービス側の拡充が進んでいます。
2. コグニティブと Bot
   この分野では、抽選枠を取り合っている Amazon Echo や、半額セールで話題になった Google Home などのスマートスピーカー分野が一気に注目を浴びたことで、その背後にある技術領域として普及の芽が出ています。私自身も Amazon Echo と Google Home mini 両方試していていますが、「ちょっとした便利」の積み重ねがなかなか良いです。
3. IoT とセキュリティ
   各クラウドベンダから IoT 機器や IoT ゲートウェイに対するデバイス管理 SDK が登場したり、IoT デバイスに対してリーチャビリティとセットで識別/認証機能を提供する SORACOM が KDDI に買収されるなど、IoT とセキュリティにまつわる戦場は隣接領域を巻き込みながら拡がっています。
   なかでも「できる人が居たから」感もありますが、AWS が RTOS ベースのディストリビューションまで踏み込んだのは面白い動きだと思います。
4. 新たなる「ラストワンマイル」競争
   SORACOM が KDDI に買収されたほかは、特に大きな動きはなかった分野でした。とはいいつつ、5G フィールドトライアルなど、地道に研究開発が進んでいる分野です。SORACOM の Sigfox 対応や、LoRaWAN による「全国サービス」を展開しようとするセンスウェイなど、普及期に向けた動きは見えつつあります。

5. エッジコンピューティング
   AWS Greengras や Azure IoT Edge SDK など環境は揃ってきています。PoC レベルの事例は増えてきているため、来年持ち越しという感じでしょうか。
6. FinTech
   まさかここまで仮想通貨が盛り上がるというかバブルになるとは思っていなかったのが正直なところです。その一方で、本丸である金融機関の API 化は、規格化など地道に進んでいますがまだまだ道半ばといったところでしょうか。
7. ユーザーインターフェースの拡張
   淡々と VR がコンテンツを増やす一年でした。やはりエロは大事ですね。年末には VR 男優の特殊技能に関する記事が注目を集めました。
   ウェアラブル分野では、Apple Watch 3 がついに単独でセルラー通信に対応したのが一番大きな動きです。この裏には組み込み型の eSIM など様々な技術革新が存在しています。eSIM はいいぞ。
8. Docker のコモディティ化
   世の中全て Docker どころか、そのオーケストレーションも Kubernetes でまとまりつつあります。
9. セキュリティ教育
   私自身も SecHack365 で関わっている NICT（情報通信研究機構）のナショナルサイバートレーニングセンターの発足など国を挙げてのセキュリティ人材育成が進んでいます。
10. おまけ
    3 月に無事本編完結した「本好きの下剋上」、なんと『このライトノベルがすごい！2018』単行本部門で 1 位でした。

ボイス UI（VUI）の躍進

OS に紐付きであった Apple Siri や Windows の Cortana から時計の針が一つ進み、Amazon Echo や Google Home、LINE Clova などスマートスピーカーとしてのデバイスが登場しました。2016 年ぐらいから拡充が進んでいたコグニティブ分野の成長との相乗効果もあり、ボイス UI が一気に花開きつつあります。

すでにプログラマブルになっていると言うこともあり、来年はその活用が進むと思いますが、その中でも「日常のちょっとした便利さの積み重ね」というのは潜在的な大きな魅力を秘めています。このようなデバイスで一番重要なことは「気づいたときには無いと困る状況」をいかに作り出すかですが、例えば料理中の音声メモなど、スマートスピーカーはそういった領域が得意に見えます。

ボイスチャットなどからスマートホームなどのボイス UI に「侵入」するという課題も明らかとなっていますが、これには「音声による識別」がまだまだ不十分という背景があります。風邪を引いて声が変わるな

ど、他の生体識別技術と比べても難しい技術分野ではありますが、重要な操作にはスマートフォンとの2FA を要求するなどのリスクベースの判断など、そういったセキュリティ上の考慮事項に関するガイドライン化が進むのでは無いかと考えています。

ボイス UI に引っ張られる形で、いわゆる IoT 家電として少しずつ製品が増えているスマートライトなどに注目が当たっているのもなかなか面白いところです。この辺は、2018 年から既存の IoT 家電スタートアップの買収劇が大いに進んでいくものと見ています。

PKI の抱える課題が顕在化

由緒正しい Verisign の流れを汲んでいた Symantec が、そのサーバ証明書事業を DigiCert に売却することになりました（2017 年 10 月に買収完了）。これはブラウザベンダーとしての Google と Symantec との争いが背景となっています。

Google は、以前より Certificate Transparency（CT）という枠組みを構築し全てのサーバ証明書に対して CT への対応を求めています。CT は証明書の誤発行や偽造を防ぐために、CA が証明書を発行するときに同時に CT にそれを登録する監査の仕組みで、RFC6962 としても標準化もされています。ブラウザは、HTTPS 等で通信するときに CT の監査ログを確認することで正しい証明書であることを確認します。この仕組みは、例えば Google Chrome であれば 2015 年 1 月より段階的に導入され、2016 年 6 月のリリースからは CT に登録されていない証明書で警告が表示されるようになっています。

ここには二つの課題があります。一つは、あの Verisign を買収した Symantec ですらも、Google の求める基準を達成することができていないということです。常時 TLS 化が進んだ結果、利用そのものが増えていることもその背景にあるかとは思います。

もう一つは、証明書という PKI の根幹に対して、ブラウザベンダーにすぎない Google の「力」が巨大になっているということです。DNS の信頼性が揺らぎ、証明書に通信の安全性を依存している現状において、証明書の偽造リスクは決して無視できず Google の圧力にも理はあります。その一方で、Symantec に対する「サーバ証明書の信頼を無効化」という強すぎるカードなど、PKI そのものに対する Google の支配力が高まっているのも事実です。綺麗な王様であるうちは良いのかも知れませんが、インターネットの信頼モデルの根幹を Google に握られつつある、という現状は知られるべきでしょう。

その一方で、Let’s Encrypt やクラウドベンダによる無料証明書によって、とにかく DV（ドメイン認証）でよければ手軽に（たとえ悪意を持つものであっても）HTTPS による暗号化ができるようになった現状は、マルウェアの被害拡大なども経て 2018 年以後より議論が深まっていくことでしょう。

Windows Subsystem for Linus

WSL は良いぞ。

Linux（Ubuntu）の amd64 バイナリがそのまま動く WSL は、まずは開発者向けリリースとして提供されている段階ですが、大きな可能性を持っています。実際に試した人はご存知だと思いますが、ターミナルが（昔に比べれば雲泥の差で頑張っているとはいえ）ショボい事を除けば、一般的なユーザアプリケーションはそのまま動くようになってきています。ターミナルも、代替のものが OSS でリリースされています。

Windows の世界観との繋ぎ込みはまだまだ道半ばですが、UNIX というか Linux っぽい作業環境のために Mac を使っているような人は一度試して見ると良いかも知れません [69]。とくにビルド環境としては Ubuntu/amd64 のバイナリを、クロスコンパイルではなく普通に作って動かす事ができます。是非はともかく、世の中大半の環境で Linux がポータブルなバイナリになりつつある現状ではかなり大きなメリットです。

2018 年にもおそらく 2 回の大きなアップデートが Windows に来ると思われますが、Linux 開発環境としての進歩が楽しみなところです。

GPU コンピューティング

いままさに NVIDIA の EULA 変更による議論が真っ盛りですが、それはさておき、2017 年から 2018 年にかけて GPU がより一般化が一段階進むタイミングと見ています。特に根拠はありませんが、おそらく向こう 2 年くらい続くであろう機械学習のカジュアル化の波に乗って、GPU コンピューティングでモノが書ける人が増加し、それを活用した事例が出てくるような気がしています。

最終的にはクラウドベンダによりサービスに消化された機能を利用するようになっていくでしょうが、そうなってもやはり「中身を知っていることは強い」ので、そういった技術者の取り合いが過熱していくことでしょう。

また、末端でのデータ集約などのエッジコンピューティングの領域でも利用が広まり、AWS や Azure が提供しているエッジ側の FaaS と相まって、一つの技術領域として確立していくのではと予想しています。

[69] もちろん、OS 自体との統合としてみるとまだまだ比較になりませんが。

NGN の課題と IPv6 の普及？

現状の NGN における PPPoE 収容先の限界によって IPv6 IPoE 接続が急速に知られる[70]ようになるなんて理由で IPv6 の普及が進むなんてことになるとは正直思っていませんでした。

様々な歴史的経緯によってたまたま既存の IPv4 の通信が遅くなっているだけの話なのですが、既に PPPoE 収容方式の改善案などは見えてきているため、2018 年にはある程度改善されるのでは無いかと考えられます。

量子コンピュータ

IBM Q に続いて Azure などから使えるおもちゃが出てきたので、来年は「触ってみた」系の記事がたくさん出てくるものと思います。というか正直この本で扱う予定だったのですが落としました。

[70] 一方で PPPoE を張り直して軽い収容先までリセマラする「NTE ガチャ」も行われている模様。

あとがき

気づいたら技術書典 5 の「TechReport 2018.10」でサークル「めもおきば」として発行した同人誌が 10 冊目で、今回の冬コミで同人活動 3 周年を迎えることになりました。これも読んでいただいた皆様のおかげです。本当にありがとうございます。

大変ありがたいことに、Hardening の体験記など既刊も手に取っていただける方がいらっしゃるのですが、既刊 6 冊ともなるとサークルスペースの都合もあるということで、今回は過去の 10 冊の中から、「Serverless を支える技術」に連なる 4 冊を除いた 6 冊を再編集するという形で総集編をだすことにしました。

これからも「アウトプットしないのは知的な便秘」をモットーに、引き続き書きたいことをやりたいように取り上げていきたいと思います。

めもおきば TechReport 総集編 Vol.1

発行日	**2018 年 12 月 30 日**	**初版 コミックマーケット 95**
	2019 年 4 月 14 日	**第 2 刷 技術書典 6**
著者	**Aki @nekoruri**	
	aki@nekoruri.jp	
発行	**めもおきば**	
	http://d.nekoruri.jp/	
印刷	**ねこのしっぽ**	