事例研究
TOMOYO Linux が考えたセキュリティ

セキュリティを決めるのは軍事力と政治ではない。
法律と裁判所でもない。
個々の人がどのように考え、行動するのかで決まる。

目次

- 自己紹介
- TOMOYO の歴史を振り返る
  - 第１部　プロジェクト開始からＯＳＳ公開まで
  - 第２部　ＯＳＳ公開からメインライン化挑戦開始まで
  - 第３部　メインライン化挑戦開始からメインライン化達成まで
  - 第４部　メインライン化達成後の出来事
- まとめ

自己紹介

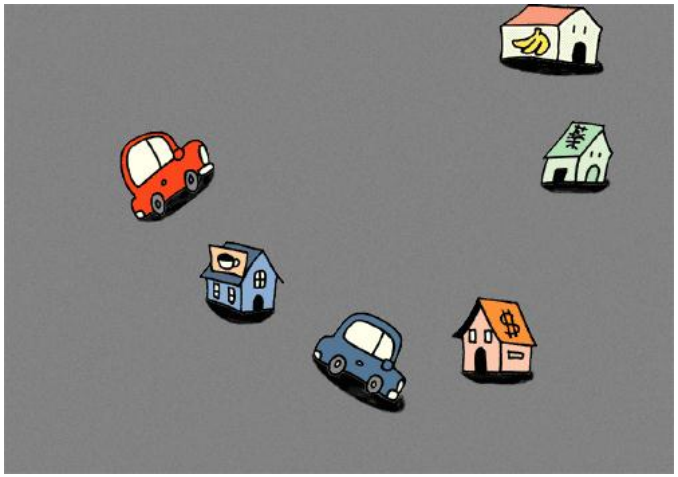
- 半田哲夫（熊猫さくら）
  - 職業プログラマ／夢追い人
  - 数学音痴／理論音痴
  - セキュリティの研究者や専門家ではない
- 今日は、TOMOYO Linux の歴史を軸に、メインライン化の苦労話や TOMOYO が考えたセキュリティについてのお話です。

TOMOYO の歴史を振り返る

第1部
プロジェクト開始からOSS公開まで

プロジェクト開始（2003.4）

- 普通のシステムは無防備であり、root 権限を奪われたら改ざんや漏えいなどやりたい放題になってしまう。
- バッファオーバーフロー攻撃による侵入防止策の模索から始まる。



SELinux に挑戦（2003.4）

- SELinux がファイアウォールのように紹介されていたので試すも、数日で挫折。
- Linux のインストールをしたことさえ無かった私には、カーネルのコンパイルは未知の世界であった。
- なんとか SELinux をインストールできたが、システム管理者経験の無い私にはポリシー（アクセス制御規則）の設定は無謀だった。

SELinux を使わないで済む方法は無いだろうか？（2003.5）

- chroot や User Mode Linux を試す。
  - root 権限を奪われても自由にできないようにすることはできないだろうか？

- 特定のiノード番号を持つプログラムの実行を禁止するカーネルを試作する。
  - 初めてカーネルの修正を行う。

- SubDomain の調査をする。

目標を改ざん防止に限定することで、ポリシーの管理を省略できないだろうか？（2003.6）

- ルートファイルシステムを読み込み専用化することに挑戦。（開発コード SAKURA Linux ）
  - 読み込み専用でよいファイルと読み書きが必要なファイルとを分離するためにはまず把握することが必要。
    - ファイルのオープンを観測するカーネルを作成。
      - カーネルの構造を全く知らなかったので、今から思えば恥ずかしいミスばかりしていた。

やっぱり改ざん防止だけでは頼りない（2003.7）

- SELinux で保護されたシステム上で非 root 権限で動作する User Mode Linux に関して、サービスごとに chroot で隔離して、それぞれの環境を読み込み専用ファイルシステム化して、さらに非 root 権限でサービスを動作させるようにできれば、ホストの root 権限を奪うのは相当困難にできるのではないか？
  - ・・・面倒すぎてやってられまへん。　Orz

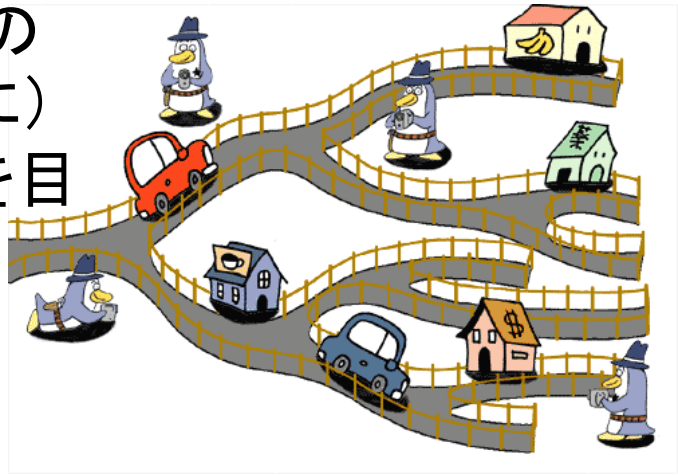
パス名でポリシーを定義できれば、だいぶ楽になるのでは？（2003.8）

- システムの動作（プログラムの実行やファイルのオープン）を観測するカーネルの開発に着手。
  - タスク構造体に手を加えてシステムの動作を追跡する方法を開発。
- 演習：TOMOYO Live CD を用いてシステムの動作を解析する。

  http://tomoyo.sourceforge.jp/download.html

追跡はできたけれども、制限もできなくちゃ意味無いよね？（2003.8）

- 観測された動作を理解し、その動作のみを認めている限り、予想外の出来事が生じないようにできる筈。
  - SELinux のポリシーをカスタマイズするのではなく、観測された動作のみを認める SELinux のポリシーをスクラッチから作成できないものか？
    - アクセス制御方式の違いにより挫折。
      - アクセス制御方式の違いの背景にあるセキュリティの考え方の違いを知ったのは、（3年後に）AppArmor がメインライン化を目指し始めてからのこと。



やっぱりSELinuxは無理！（2003.11）

- SELinuxのポリシーを生成できなくても、システムの動作を追跡するカーネルをシステムの動作を制限するカーネルに改造すれば良いのでは？
  - SELinuxのポリシーを生成することを目標とせずに、独自のポリシーに基づいて独自のアクセス制御を行うカーネルを目指すことになる。

ということで・・・。（2003.12）

- SELinux を使わずに独自のアクセス制御を行うカーネルの開発に着手。（開発コード TOMOYO Linux ）

- 演習：TOMOYO Live CD を用いてシステムの動作を制限する。

ケロちゃん（開発コード CERBERUS ）登場（2004.1）

- ユーザが理解できる形でシステムの動作を制限できるようになったことで、シェルセッションでの操作を制限できるようにもなった。シェルセッションでの操作を制限できるということは、ログインシェルからの操作も制限できるということ。だから、ログインシェルで追加のユーザ認証をしてやれば、ブルートフォース攻撃への対策にもなるよね？

セキュリティスタジアム2004に参加（2004.11）

- SAKURA Linux と TOMOYO Linux を防御側で出展。
  - SAKURA Linux だけでは不十分だが、TOMOYO Linux は有効であることを確認。
- TOMOYO Linux では root パスワードを公開して攻撃してもらったが、ケロちゃんチェックにより攻撃側をヘトヘトにさせることに成功。
  - バッファオーバーフロー攻撃による侵入防止策の模索から始まったプロジェクトは、ユーザがカスタマイズ可能なポリシーを実現することにより、バッファオーバーフローで制御を失った場合に限らず、不正なログインやOSコマンドインジェクションなどで制御を失った場合も含めて、動作パターンを制限する用途で利用できるように進化した。

TOMOYO Linux がオープンソースで封印解除される（2005.11）

- 商標取得、特許調査などで時間がかかったが、お蔵入りは免れた。
- 機能面よりも名前で大騒ぎになる。

TOMOYO の歴史を振り返る

第2部

OSS公開からメインライン化挑戦開始まで

AppArmor が公開される（2006.1）

- パス名に基づいてアクセス制御を行うという点で、TOMOYO Linux と瓜二つ。
  - 尤も、TOMOYO は AppArmor の前身である SubDomain の影響を受けているので当然と言えば当然。
- TOMOYO は機能強化／使い勝手の向上を続ける。
  - メインラインという言葉を知らないまま、機能や使い勝手の向上を重ねる。

AppArmor がメインライン化の挑戦を開始する（2006.4）

- 当時は SELinux だけがメインライン化されており、「 SELinux でどんなセキュリティも実現できる」という考え方が支配していた。そのため「不完全なセキュリティで利用者を誘惑する元凶である Linux Security Module Framework 自体を削除してしまおう」という話も出るほどであった。
  - そんな中、AppArmor が「 AppArmor も LSM フレームワークを使っているんだから、削除しないでくれ～」とメインライン化への活動を開始した。
    - ところが・・・。

宗教戦争（2006.5）

- 考え方の違いにより、歴史に残る大論争が展開された。
  - SELinux の主張：ファイルのリネームなどによりアクセス可否の結果が変化してしまうようでは一貫した判断ができない。リネームなどの影響を受けないようにするためにパス名ではなくラベルを使うべきだ。
  - AppArmor の主張：難しすぎて無効化されてしまうようなセキュリティよりかはマシだ。
- 2003年にパス名を用いて観測した結果から SELinux のポリシーを作成しようとして挫折した原因が明らかになった。
  - SELinux ではリネームなどの影響を回避するために、パス名を使わない設計を採用していたのである。

SELinux が考えたセキュリティって何？

- 情報を隔離し、情報がどのように流通するかを制御できることが重要。
  - TCSEC や CC などの流れを汲んでいる。
  - MLS/MCS などを実現する上で、ラベルに基づくアクセス制御が不可欠。
- ラベルを使わないセキュリティなんてありえない。

TOMOYO の歴史を振り返る

第３部
メインライン化挑戦開始から
メインライン化達成まで

運命の CE Linux Forum "Japan Technical Jamboree12"（2006.12）

- AppArmor がボコボコにされていくのを横目に、TOMOYOはメインライン化に挑戦する気も、LSMフレームワークに対応する気もさらさらなかった。
  - 当時のLSMフレームワークの仕様では、ルートディレクトリからマウントポイントまでのディレクトリ名部分を導出できないという制約があった。
- しかし、組込み Linux 開発者向けの勉強会で TOMOYO のデモを行った際、「我々はメインラインに入っていないものは使わない」と言われ、メインライン化に挑戦することを勧められた。
  - そして、４月にある CELF 国際会議で TOMOYO の発表を勧められた。

運命の第72回 YLUG カーネル読書会（2007.2）

- TOMOYO がボコボコにされた。
  - ただし、「こんなもの使えねぇ」という批判ではなく、「これは何が何でもメインラインに入れろ」という激励の意味で。
  - http://video.google.com/videoplay?docid=-5205950078661881140
- ・・・というわけで、ここでもメインライン化に挑戦することを勧められ、TOMOYO もメインライン化に挑戦することになる。

CE Linux Forum Worldwide Embedded Linux Conference 2007 にて（2007.4）

- TOMOYO の海外デビュー。

- ここでも Linux Weekly News の Jonathan Corbet さんから「メインライン化に挑戦すべきだ」という激励が。

TOMOYO Linux もメインライン化の挑戦を開始する（2007.6）

- Ottawa Linux Symposium 2007 での BoF で TOMOYO Linux の発表をすることになり、それに先立ち最初のメインライン化提案を行う。
  - パッチのコーディングスタイルなどの流儀を知らずにいろいろミスばかりしていた。

Ottawa Linux Symposium 2007 にて（2007.6）

- SELinux 関係者と AppArmor 関係者も参加する中、TOMOYO のデモを行う。
- LSMフレームワークのメンテナである Chris Wright さんが「 TOMOYO と AppArmor の両方を入れることはしないだろう」と発言。
  - これがきっかけで「ラベル対パス名」という競争に加えて、「 TOMOYO 対 AppArmor 」という競争も始まった。
  - システム全体をカバーできるか否か、ユーザが内容を理解して設定できるか否か、細かいところに手が届く気配りがされているか否かなど、TOMOYO のほうが丁寧に設計されているのだが、パス名ベースである時点で大差ないと思われてしまっていた。

どうやって AppArmor との差別化をした？

- 実は、AppArmor は途中でメインライン化を諦めてくれたので、TOMOYO と AppArmor の間の競争は自然に消滅してくれた。
  - そのため、TOMOYO の意義を主張することに専念できるようになった。
- その後、ラベルでは扱えないセキュリティも存在することを説き続けた。
  - SELinux の主張：ファイルのリネームなどによりアクセス可否の結果が変化するのは問題だ。
  - TOMOYO の主張：ファイルのリネームなどによりシステムの動作やファイルの使われ方が変化するのも問題だ。アクセス可否の結果が変わらないだけでは不十分だ。

TOMOYO が考えたセキュリティって何？

- OSレベルのアクセス制御が情報の流通経路を制御できるだなんて発想は無かった。
  - 情報を格納する容器であるファイルにラベルを割り当てることはできるが、情報そのものにはラベルを割り当てることができない。
- 情報へのアクセスを認められたユーザが、どのような行動をするかに依存している。
  - Twitter とかに書き込んでしまうかもしれないし、メールで送信してしまうかもしれない。あるいは、ファイル名という形に変換して故意にリークさせるかもしれない。
- つまり、アプリがどのように情報を扱うかという側面を無視して、OSレベルのアクセス制御だけでセキュリティを語るというのは無茶。

それってアプリレベルの セキュリティの担当でしょ？

- 確かにそうだけど、アプリレベルでのアクセス制御の内容はバラバラである。1か所でも漏れがあったり、バッファオーバーフローなどで制御を奪われたら即死。
  - カーネル内でベースラインとしての制御を行おう。
- 本来はOSレベルのアクセス制御が扱うべき話ではないが、アプリレベルでのアクセス制御を補強するために、TOMOYOはOSレベルのアクセス制御として、カーネル内でパス名に基づくアクセス制御を持ち込んだ。
  - しかし、パス名を導出できるようにするためにはLSMフレームワークの修正が必要であり、TOMOYO や AppArmor は理論面だけでなく実装面でも不利であった。

SMACK の登場（2007.7）

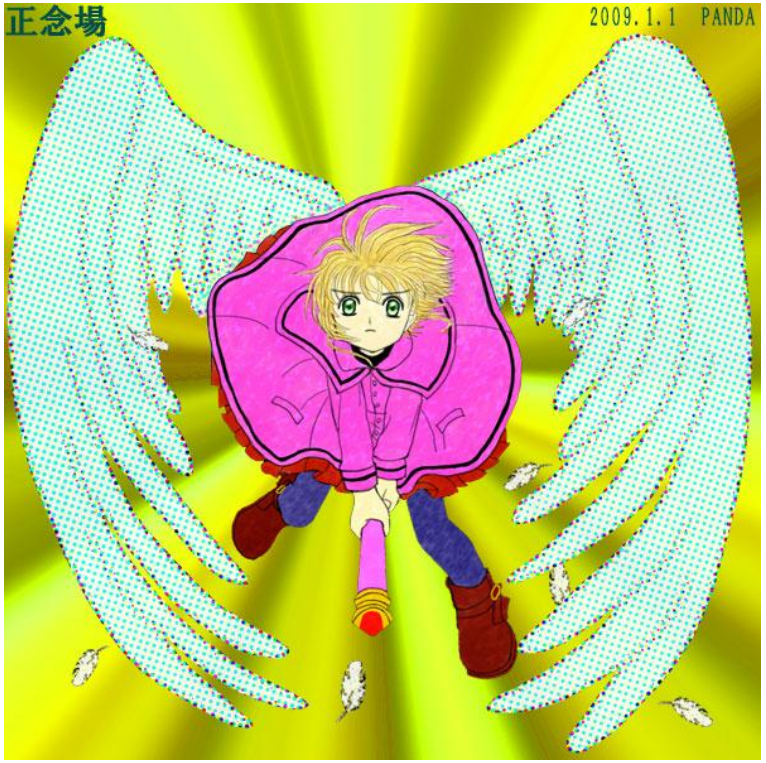
- LSMフレームワークの存廃やあるべき姿を巡る議論の中、SELinux のサブセットのような、ラベルに基づくアクセス制御機構が提案された。
  - LSMフレームワークの変更を伴わずに実現できることや、Linus Torvalds さんの「LSMは無くさせないぞ！」発言があったことから、Linux 2.6.25 でメインライン化された。

なかなか進展が無い状況が続く

- パス名ベースのアクセス制御を行うのに必要なLSMフックの修正が受け入れられず、先へ進めずに焦ったり落ち込んだりする日々。
- 業を煮やしてLSMフックを使わないバージョンを提案して怒られたりもした。
- 誰がどのサブシステムのメンテナ（責任者）であるかを把握していなかったため、誰の意見に従うべきかが判断できない部分もあった。

でも、遂に動き出す（2009.1）

- TOMOYO がカーネル内でパス名を導出するために必要なLSMフックが追加される。



LinuxConf.au 2009 後（2009.2）

- 理論面ではラベルでは扱えないセキュリティも必要であることを説明し続け、実装面では機能の絞り込みとコーディングスタイルの修正を続けた結果、15回目の提案で遂に Linux 2.6.30 に入れてもらえる内定をもらう。

メインライン化達成（2009.6）

- ただし、メインライン化されることを最優先したため、最低限の機能しか使えない。デモレベルの機能しか備えておらず、実用に耐えるレベルには程遠い。

TOMOYO の歴史を振り返る

第4部
メインライン化達成後の出来事

メインライン化記念パーティ（2009.7）

- これはOSSの歴史に残るイベントだったかも？
  http://sourceforge.jp/projects/tomoyo/wiki/ThankYou

ディストリビューションへの採用が始まる

- Debian Squeeze ／ Ubuntu 9.10 ／ ArchLinux ／ Mandriva ／ openSUSE 12.1 などでメインライン版の TOMOYO が採用されていく。
- しかし、SELinux を推進する Fedora および RHEL では採用されない状態が続く。

AppArmor がメインライン化への挑戦を再開する（2009.9）

- 2007年時点では競争相手だったが、今となっては共創相手になった。
- 徹底的に実装面のレビューに注力し、有用性や概念の論争をさせる隙を与えないようにした結果、AppArmor も Linux 2.6.36 で入ることができた。

独自拡張版の改良はその後も続く

- 独自拡張版（ TOMOYO 1.x ）の機能や使い勝手を向上させながら、メインライン版（ TOMOYO 2.x ）へ移植するための提案／交渉（試行錯誤）を続ける。

AKARI の登場（2010.10）

- 予想通り、RHEL6 では TOMOYO が有効にされなかった。
- カーネルパッケージの置き換えは利用者の精神的負担が大きいことから、ローダブルカーネルモジュールとして追加するだけで独自拡張版の機能の主要部分を使えるバージョンを開発して公開する。

現在（2011.8）

- メインライン版は、ネットワークに関するアクセス制御機能は未搭載であるものの、アクセスログ取得機能などを備え、Linux 3.1 で実用に耐えるレベルになる予定。
- SELinux と AppArmor の論争前は「 TOMOYO は SELinux を使いやすくしたもの」と思っていたが、論争後は「 TOMOYO は SELinux の不足を補うもの」という考えに変わり、TOMOYO の必要性の主張と並行して、「ラベルに基づくアクセス制御だけでも、パス名に基づくアクセス制御だけでも不十分。だから、複数のLSMモジュールを同時に有効にできるようにすべき」という主張も続けている。

まとめ

考え方の違い

- SELinux の考えたセキュリティ
  - セキュリティを決めるのはポリシーとカーネルの正しさである。
  - 抽象化して数学的に扱えるもの
    - 数式で説明できるものを得意とする

- TOMOYO の考えたセキュリティ
  - セキュリティを決めるのはプログラムの動作やユーザの操作である。
  - 抽象化して数学的に扱うことができないもの
    - 数式で説明できないものを得意とする

歴史的な経緯

- アクセス制御というセキュリティには役割分担が存在する。
- OSレベルのアクセス制御
  - 主体が客体にアクセスできるかどうか
    - WebコンテンツファイルやDBファイル
  - 主体の意図には関知しない
- アプリレベルのアクセス制御
  - 主体が客体に含まれている情報をどのように扱うか
    - Webサーバはファイルの内容をWebコンテンツとして解釈し、DBサーバはファイルの内容をDBとして解釈する。
  - 主体には意図が存在し、意図を考慮した制御が行われる。

TOMOYO の経緯（1）

- 当初はバッファオーバーフローで制御を失った場合の被害を軽減することが目標だった。
  - バッファオーバーフローで制御を失っても、自由にはさせないようにするためにカーネル内でアクセス制限を行おうとした。許可する動作パターンとして、プログラムの実行履歴（ fork/execve ）を利用した。
- その後、ブルートフォース攻撃で侵入された場合の被害を軽減するという目標も加わった。
  - バッファオーバーフローで制御を失った場合と、不正なログインやOSコマンドインジェクションなどで制御を失った場合とを区別せずに、動作パターンを制限するという用途で利用できるようになった。

TOMOYO の経緯（２）

- さらに、情報へのアクセスの可否を制御するだけでなく、情報の所有者が望む形で正しく情報を使ってもらうという目標も加わった。
  - そのために、システムの動作や情報の使われ方に影響を与える様々なパラメータ（使ってよいファイル名、コマンドライン引数や環境変数などといった、ラベルに基づくアクセス制御では扱えないもの）もチェックするようになった。
- これらにより、TOMOYO Linux は「主体の意図を考慮したアクセス制御をカーネル内で行うアプリケーションファイアウォール」としての側面を備えるようになった。

TOMOYO のコンセプト

- 一挙手一投足をコントロールし、予想外の出来事が生じないようにしたい。
  - そのために、全ての動作を記録し、その内容を確認して編集し、納得のいく動作だけを認めるようにする。
- プロジェクトページの説明文より:「TOMOYO Linux はシステムの振る舞いに注目します。プロセスは何かの目的を達成するために生成されます。TOMOYO Linux は（出入国審査官のように）それぞれのプロセスに対して目的を達成するのに必要な振る舞いや資源について報告させることができます。また、保護モードを有効にすることにより、（運用監視人のように）システム管理者により承認された振る舞いと資源へのアクセスのみを許可することもできます。」

時代状況の変化

- TCSECが考案された時代とは違う。
- ネットワーク接続が当たり前に
  - アプリのセキュリティ＝目立つ存在
  - OSのセキュリティ＝目立たない存在
- 攻撃手法の非バイナリコード化／アプリ動作の複雑化
  - バッファオーバーフロー攻撃への対策は進んだ。
    - スタックガード／アドレス空間のランダム化／CPUのNXビットなど
  - ソーシャルエンジニアリング攻撃の増加
    - フィッシング詐欺

今あるOSでできるセキュリティ強化

- 例えば、HTMLを使わないことでコードとデータの分離を確実にすることで、多くのセキュリティ問題は解決できる。
  - http://Kumaneko-Sakura.sblo.jp/article/288772.html
- メリットはセキュリティ強化だけでない筈。
  - ブラウザに代わる選択肢（日経システム構築2003年7月号）
  - Webアプリのトラフィックは99%削減可能（私の経験）
  - ブラウザのバージョンアップへの追随コスト
    - IE使いますか？ブラウザ捨てますか？（とある個人の欠陥覚書 2011.07.21 ）
- OSレベルのアクセス制御強化という新しい道具の登場
  - システムにログインさせることを恐れないで。

私がこれからのOSに期待していること

- システムの見える化がますます重要に
  - でも、既存の役割分担ではOSで対処できる範囲は限られている。
    - 例えば、表示可能領域の制限とか
      - 信頼できるタスクバーやタイトルバーがあれば、フィッシング攻撃に騙されにくくできる。
      - 既存のOSが提供するAPIでは、そういうことを実現できていない。

従来からの役割分担のままで良いのだろうか？

- 縦割りなセキュリティでは、最も脆弱な場所から攻撃される。

- 今回は「セキュアなOSを作ろう」というコースなので、従来からの役割分担に囚われないOSを開発するというのも有り？

車輪の再発明は無駄か？

- そうとも限らない。時代背景が違うだろうから。
  - 過去に良くないと思ったことは二度と試そうとしない、というのはセキュリティ専門家が陥る落とし穴である。それでは、セキュリティ専門家以外でも使えるものを提供できない。
- TOMOYO は既存のＯＳではタブー視されてきたことを再発明した。
  - セキュリティ専門家以外でも使えることを重要視している。
- 発明を利用するだけの人には得られないノウハウがある。
  - 遠回りを経験して初めて、道筋の良し悪しが判断できるようになる。

余談・・・私が会社員生活で感じたこと

- 新人時代にチャンスが訪れる。「新人だから失敗しても仕方が無い」と思ってもらえるから、図々しくチャレンジするチャンスだと思う。
  - その時に自信を持って意見を言えるように、自分の考えを持っていてほしい。
    - その時の判断が、行動が、独自のアイデアを形にしていく。

最後に

- 真摯さについて・・・ドラッカーではないけれど。
- セキュリティと称して暗号化だのアクセス制限だのといった、省エネに逆行することは本当はやりたくない。
- 私はCCさくらのような世界が大好きである。侵略される方が悪いと考える人もいることは承知している。でも、そういう考えではない人を無理矢理巻き込むのは勘弁してほしい。それは、足の引っ張り合いであり、破滅への道であるから。
- セキュリティとは、人の欲望が生みだした、永遠に続く戦争である。誰かが欲望を満たすことは、他の誰かを犠牲にすること。
- だから、私は、侵略される方が悪いという考え方を否定していく。人の考え方を変えることはすぐにはできないだろうけども。真摯にやっていくことが、戦争を終わらせることのできる、本当のセキュリティであると信じている。