Network Bound Disk Encryptionとは

レッドハット株式会社

2018-09-25

森若 和雄 <kmoriwak@redhat.com>

このスライドの位置付けと目的

- 対象
  - セキュリティ向上のためストレージ暗号化と、自動的に暗号化解除するNBDEを活用したい人
- 目的
  - RHEL 7.4から提供が始まったNetwork Bound Disk Encryption(NBDE)の概要紹介
  - NBDEが解決するストレージ暗号化の問題と、解決しない問題を把握する

Copyright Red Hat K.K. All rights reserved.

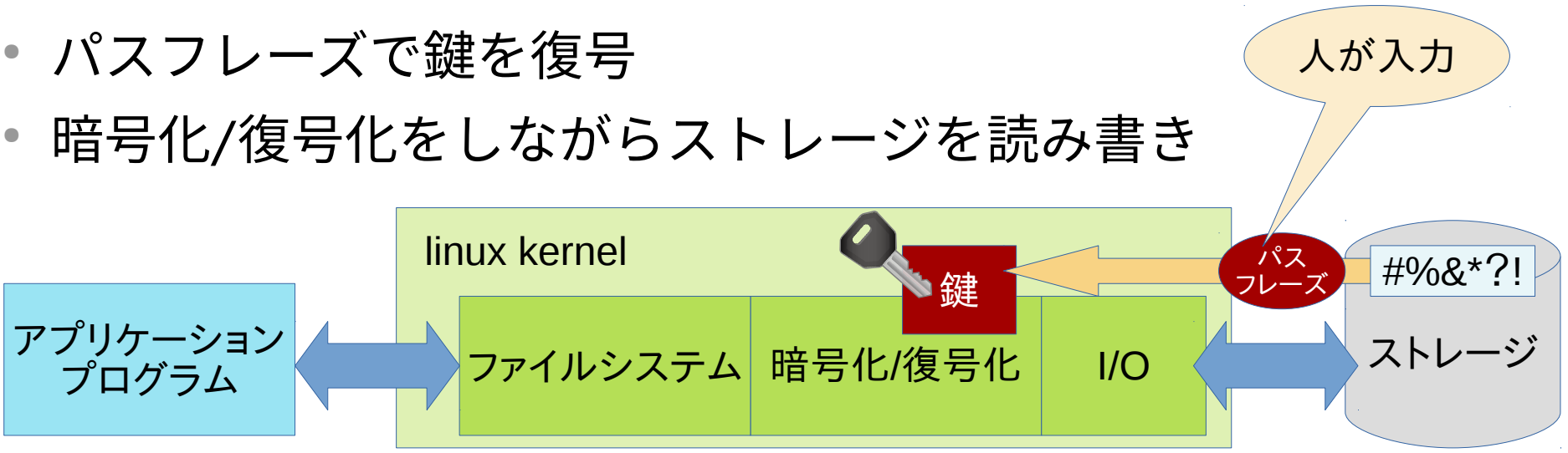
概要

- 通常のストレージ暗号化
- Network Bound Disk Encryption(NBDE)とは
  - tangサーバとは?
  - clevisとは?
  - tangサーバをどこに置くか?
- RHELでNBDEを利用する

Copyright Red Hat K.K. All rights reserved.

通常のストレージ暗号化

- LUKSストレージ暗号化
  - ストレージに暗号化された鍵を保存
  - ユーザーがパスフレーズを入力
  - パスフレーズで鍵を復号
  - 暗号化/復号化をしながらストレージを読み書き



Copyright Red Hat K.K. All rights reserved.

ストレージ暗号化の目的

- ストレージを紛失しても情報が漏れないようにする
  - 想定している問題: 物理的盗難、移動中の紛失、不適切な廃棄 etc.

※目的ではない(実現されない)こと:

- システムに侵入されてデータを盗まれることの予防
- 特定のアプリケーション以外からはデータを参照できないように制限する

Copyright Red Hat K.K. All rights reserved.

ストレージ暗号化の課題

- 「ストレージを紛失しても情報が漏れない」ようにするため、ストレージの中に鍵そのものは置けない
  → 最初に人間がパスフレーズを入力する
- 問題になるケース
  - システム台数が多い場合
  - 負荷に応じてシステムを増減させたい場合
  - その他短い時間でストレージを使いはじめたい場合

Copyright Red Hat K.K. All rights reserved.

Network Bound Disk Encryptionとは

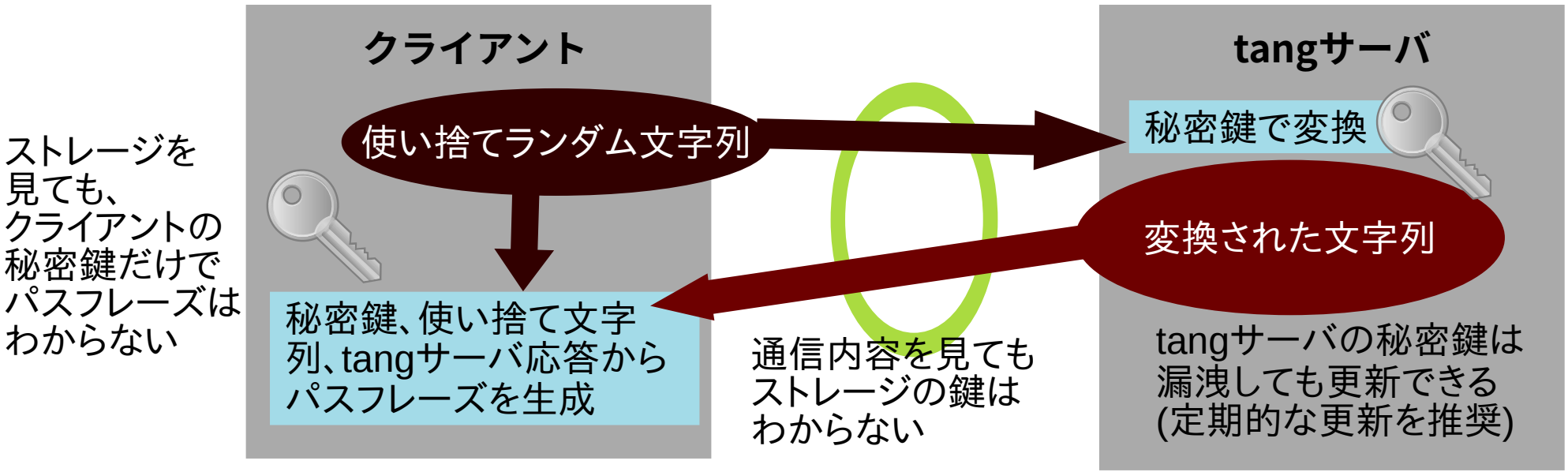
- ストレージのパスフレーズを自動的に生成する
  - 「特定のサーバ(tang サーバ)群」にアクセスできる
    =パスフレーズを生成できる
    =自動的に暗号化を解除できる
  - 冗長化のため複数のtangサーバを利用して、あらかじめ決めた台数以上にアクセスできた場合に鍵を生成
    - それぞれのtangサーバは独自の鍵を持ち共有しない

※ストレージとtangサーバ秘密鍵の両方が漏洩すると暗号化解除される

Copyright Red Hat K.K. All rights reserved.

tangサーバとは?

- 何か文字列を送ると秘密鍵で変換してから返信するだけ
- クライアントでさらに変換してパスフレーズを生成



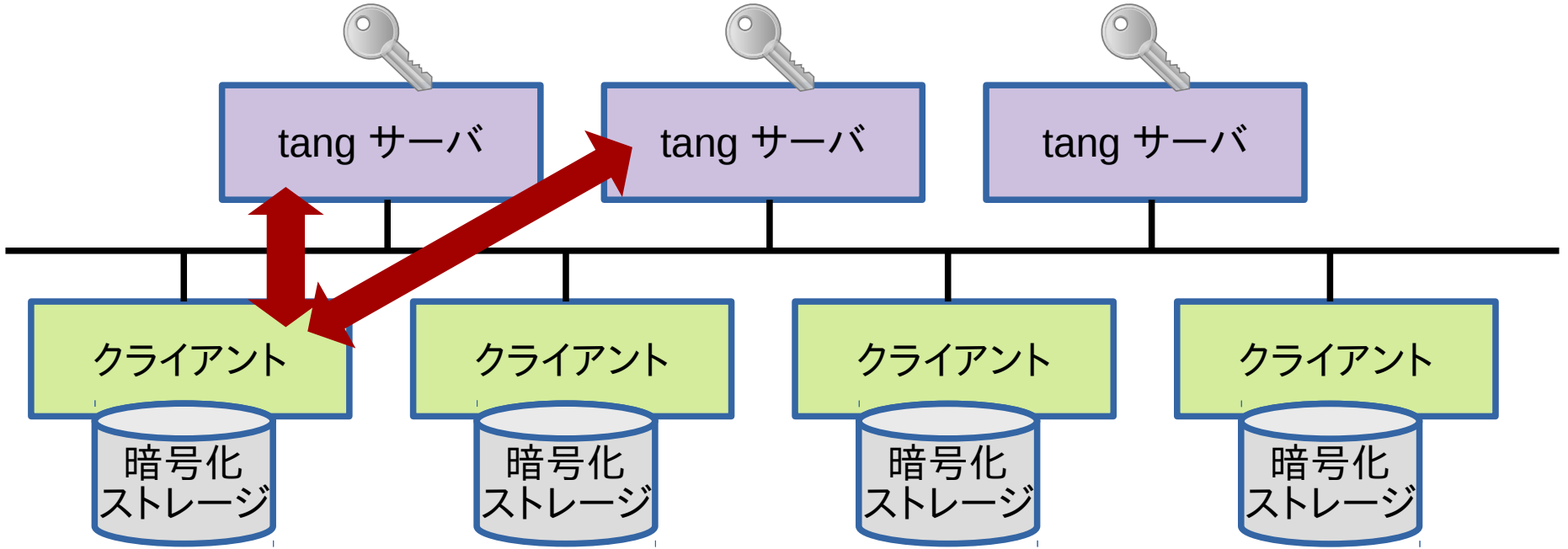
Copyright Red Hat K.K. All rights reserved.

clevisとは?

- 複数のtangサーバ、TPMなどを組み合わせて利用する仕組み
- 例:
  - tangサーバ2台以上に接続できると自動的にパスフレーズを生成できる
  - tangサーバ2台以上に接続できるかTPM2.0で保存された鍵があれば自動的にパスフレーズを生成できる

Copyright Red Hat K.K. All rights reserved.

NBDEイメージ図



たとえば2台以上のtangサーバにアクセスできると復号できる

Copyright Red Hat K.K. All rights reserved.

tangサーバはどこに置くか?

- tangサーバの秘密鍵と、clevisが管理するメタデータの両方が揃うとストレージの暗号化を解除できる
- 仮想化を利用する場合にtangサーバの鍵と暗号化ストレージが同一のストレージに配置されないようにする
  - そもそも暗号化した目的は「ストレージを紛失しても情報が漏れないようにする」こと
  - 別のデータセンタに置いてVPNで接続する等

Copyright Red Hat K.K. All rights reserved.

RHELでNBDEを利用する

- tangサーバ側

```
# yum install tang
# systemctl enable --now tangd.socket
```

- 暗号化ストレージ利用側

```
# yum install clevis-luks clevis-dracut clevis-systemd
# clevis luks bind -d /dev/sdX tang \
'{“url”:”http://tang.example.com”}'
```

→ tangサーバのキーを確認し、既存のパスフレーズを入力

```
# dracut -f
```

※ root fsを暗号化する場合DHCPではなく固定のネットワーク設定がdracutで必要。

※ root fs以外を暗号化する場合 /etc/crypttab で _netdev 指定をおこなう。

Copyright Red Hat K.K. All rights reserved.

Thank You

Copyright Red Hat K.K. All rights reserved.

関連資料

- RHEL 7ドキュメント「Security Guide」内「Network Bound Disk Encryption」
  - https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/security_guide/sec-using_network-bound_disk_encryption
- clevis
  - https://github.com/latchset/clevis
- tang
  - https://github.com/latchset/tang
  - 理論的な背景であるMcCallum-Relyea exchangeの説明

Copyright Red Hat K.K. All rights reserved.

想定Q&A(1/2)

- 今はストレージを暗号化していないけど後からin-placeで暗号化するように変更できる?
  - できません
- 既存パスフレーズは使えなくなるの?
  - clevisは既存のパスフレーズの他に自動暗号化解除を使えるように構成します
- 暗号化するとパフォーマンスは落ちる?
  - はい。暗号/復号のためCPUが消費されます。ストレージI/Oが多い場合は事前に影響を確認してください。

Copyright Red Hat K.K. All rights reserved.

想定Q&A(2/2)

- tangの鍵の更新ってどうするの?
  - tangサーバで新しい鍵を作成
  - クライアントでclevis bind luksを再度実行
  - 古い鍵を使うクライアントがなくなったあとtangサーバから古い鍵を削除

Copyright Red Hat K.K. All rights reserved.

(おまけ)tangサーバの仕組み

- 登録　　クライアント側　　tangサーバ側

クライアントの秘密A

ランダムなパスフレーズKを生成
Kを使いストレージ暗号化
$a=g^A$, b, $k=Kb^A$を保存
AとKを破棄

サーバの秘密B
← $b = g^B$ を送信

- 回復

使い捨てのXを作り $x = ag^X$ を送信 →

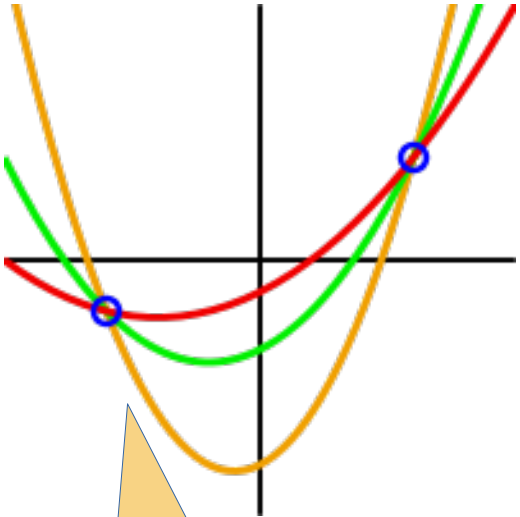
$k \div (x^B \div b^X)$ でKを回復

← x^Bを送信

Copyright Red Hat K.K. All rights reserved.

(おまけ)複数tangサーバによる回復

- 「シャミアの秘密分散法」を利用
  - パスフレーズを回復するために必要なサーバ数をk台とする
  - 関数 $f(x)=a_0+a_1x+a_2x^2+\ldots+a_{k-1}x^{k-1}$ として、パスフレーズ回復に使う秘密をa_0とする。a_1からa_{k-1}は乱数
  - tangサーバから回復した鍵がそれぞれf(x)上の異なる点(ただしx!=0)を示すよう構成する
  - k-1次関数上のk個の点が確定するとf(x)が確定し、$f(0)=a_0$を求めることができるので秘密が回復できる



3点を通る2次関数は一意に定まるが、2点しかわからないと関数は定まらない

Copyright Red Hat K.K. All rights reserved.