パスワードの話

- 自己紹介
- パスワードの保存
- パスワードの話題いろいろ
- まとめ

TODO: s5での体裁

自己紹介

- ECナビ システム本部 春山征吾 @haruyama
- セキュリティ
  - OpenSSH（本x2, OpenSSH情報）
  - 暗号技術大全
    - 18章(ハッシュ), 20章(電子署名)翻訳担当
- 全文検索システム Apache Solrの勉強会開催

資料

- 本資料
  - http://haruyama.github.com/445_14_20101211/
    - http://bit.ly/fJP5du
    - http://goo.gl/fKzJ5
  - PDF版
    - http://haruyama.github.com/445_14_20101211/ind

本講演の経緯

- 第1回神泉セキュリティ勉強会 にて，
  パスワードの保存(10分)の話を講演
- @ikepyon さんから講演依頼
  - 時間は1時間
- まずご要望を満たすために
  「パスワードの保存」の話をして，
  その後その他のテーマの話をします．

パスワードの保存

- パスワード保存の常識(?)
- Unix的パスワード保存
  - 概要
  - ハッシュ
  - Salt
  - Stretch
- Webシステムでのパスワード保存

パスワード保存の常識(?)

パスワードの保存は,

「saltを付けてハッシュしろ」

とよく言われている.

保存

- saltを付けてハッシュ化
  - 保存された情報からはパスワードは復元困難

照合

- 入力値にsaltを付けてハッシュ化. 保存情報と照合

TODO?: 図

Unix的パスワード保存

GNU/Linuxの場合

- /etc/shadow にパスワード情報を保存

形式

idsalt$hashed

例

$6$3d1ahu0b$KiH....(略)

- id: ハッシュ(後述)の識別子
  - 1 => MD5, 5 => SHA-256 6 => SHA-512
- salt: ソルト, お塩
- hashed: ハッシュ化されたパスワード情報

ハッシュとは?

暗号学的ハッシュ関数 - Wikipedia より(一部変更)

- 与えられたメッセージに対してハッシュ値を容易に計算できる。
- ハッシュ値から元のメッセージを得ることが事実上不可能であること。
  - 一方向性
- 衝突耐性 を持つこと
- 例: MD5, SHA1, SHA-256,512

salt(ソルト, お塩)とは?

ハッシュの値をかきまぜる「お塩」.

なぜ salt は必要なのか

TODO: 説明追加

- レインボーテーブル
  - ハッシュ値から平文が得られるテーブル
- 例
  - Free Rainbow Tables » Distributed Rainbow Table Generation » LM, NTLM, MD5, SHA1, HALFLMCHALL, MSCACHE
  - Ophcrack
    - レインボーテーブルを利用した Windowsのパスワードクラックツール

レインボーテーブルを利用したMD5

（デモ）

TODO: 説明追加

なぜ salt はユーザ毎に違う必要があるか

- ユーザに共通のsaltを用いると
  同じパスワードを利用する人に対して
  同じパスワード情報が生成されてしまう
- ユーザごとに異なる必要がある
  - ランダムでなくてもよい

saltのサイズ

- 伝統的なunix: 12bit
- 現在のGNU/Linux: 96bit
- CRYPTOGRAPHY ENGINEERING: ハッシュのサイズ

実際の処理

- CRYPTOGRAPHY ENGINEERING p304 の方式

PHP風の言語で記述

```
$x = '';
for($i = 0; $i < $iter; ++$i) {
  $x = hash($x . $password . $salt);
}
```

- [crypt() アルゴリズム解析（MD5バージョン）]

どちらも ハッシュを繰り返し利用している(stretch)

stretchとは?

- ハッシュを繰り返し利用することで,
  ハッシュ値を求めるのに必要な時間を増大させる
  - 攻撃に時間がかかるようになる
    - 実質的にパスワード文字数を伸ばす
      (stretchする)効果
- どれくらい繰り返されているか
  - crypt() MD5の場合: 1000回
  - crypt() SHA-256,512の場合:(デフォルト)5000回
  - CRYPTOGRAPHY ENGINEERING での例:
    2^20(約100万)回

stretchの効果(1)

stretchの効果をはかるために，PHPの hash 拡張で SHA-256を繰り返し呼ぶコードを用いた計測をした

- 方式は CRYPTOGRAPHY ENGINEERING のもの
- パスワード 10byte
- salt 32byte
- CPU 1コアのみ利用

Intel(R) Core(TM) i7 CPU 860 @ 2.80GHz で 1秒に SHA-256を約50万回計算できた.

stretchの効果(2)

・パスワードの文字種を64種とすると

文字数	総パスワード数
n	64^n
3	26万
4	1677万
5	10億
6	687億
7	4兆
8	281兆

stretchの効果(3)

1CPU(8コア)のPCでパスワード解析する場合を考察

- 1日3456億回 計算可能
  - stretch がないと...
    - 6文字が 0.2日，7文字が 13日
  - 1000回 stretch すると
    - 1日3.5億回パスワードを計算可能
    - 5文字が 3日，6文字だと 199日

stretchの効果(4)

MD5だと... Intel(R) Core(TM) i7 CPU 860 @ 2.80GHz で 1秒に 約140万回計算できた.

- SHA-256の約3倍速い
  - (私のPCでは)同じ回数stretchしても3倍弱い
- stretchの強度は,（回数）x（1回あたりの実行時間）で考えなければならない

方式の保存

現在は問題なくても，将来問題になるかもしれない

- ハッシュ関数自体
- ハッシュ化の方法
- stretch回数

長く運用するシステムでは，パスワード保存方式をパスワード情報と共に保存する必要がある．

なぜUnixはこの方式なのか?

- なぜ可逆な暗号化ではないのか?
  - 鍵を管理するのが難しい.
    - 以下からパスワード情報と鍵が漏れるかもしれない
      - バックアップファイル
      - システムの脆弱性
      - 別のOSでブート
      - 物理的な攻撃

Unix的パスワード保存まとめ

- パスワードはハッシュ化して保存
  - この時 salt と stretch を利用
- メリット
  - 鍵管理が不要
  - 生パスワードを復元できない
- デメリット
  - 弱いパスワードが記録された情報だけで破れる

Webシステムでは?

- 通常WebサーバとDBサーバは物理的に分離されている
  - Unixよりもパスワード情報と鍵が
    共に漏洩するリスクは低いだろう.
  - ちゃんとした暗号方式と鍵を利用すれば,
    攻撃者が鍵を入手できない場合 鍵の強度 ==
    パスワード情報の強度となる
    - パスワードの長さに関係ない
  - もちろん, 鍵管理のコストは無視できない
    - 漏洩, 改竄, 紛失....

TODO: 図

Webシステムでのリスク

パスワード情報の保存に関するリスクのみ挙げる

- SQLインジェクションなどによる
  パスワード情報の漏洩
- バックアップファイル，実サーバ，
  廃棄サーバなどからのパスワード情報の漏洩
- 開発者/運用者によるパスワード情報の漏洩/悪用

鍵を用いる場合の手法案

- 共通鍵暗号
- ハッシュ+暗号
- 鍵付きハッシュ

共通鍵暗号

共通鍵暗号をハッシュとして用いる
パスワード保存法もあるが,
ここではパスワード情報を暗号化する場合を考察

- メリット
  - ちゃんと暗号化し鍵が安全ならば,
    弱いパスワードもパスワード情報だけでは破れない
- デメリット
  - 鍵があればパスワードを復元できる
  - 鍵の管理の必要がある

ハッシュ+暗号

Unix的にハッシュ化したあとで暗号化

- メリット
  - ちゃんと暗号化し鍵が安全ならば,
    弱いパスワードもパスワード情報だけでは破れない
  - 鍵を保持するものでも生パスワードを復元できない
- デメリット
  - 鍵の管理の必要がある

鍵付きハッシュ(1)

鍵情報とパスワードを組合せてハッシュ

- saltの一部を鍵に?
  - 単純に鍵と平文を文字列連結をしたものをハッシュするMAC(メッセージ認証コード)は期待通りの強度がないという論文

    On the Security of Two MAC Algorithms
- hash($key . $salt . $password)
  などはMACとして用いないほうがよい.
  - パスワード保存の場合では関係ないと思われるが,あえて利用する理由はない

鍵付きハッシュ(2)

- HMACには前述の問題はない
  - CRAM-MD5はHMACを元にした
    パスワード情報保持をしている.
    - チャレンジレスポンス認証用の情報保持なので,
      応用していいかは不明

鍵付きハッシュ(3)

- メリット
  - ちゃんとしたアルゴリズムを用いて鍵が安全ならば，弱いパスワードも記録された情報だけでは破れない
    - 「ちゃんと」しているかは「ちゃんと」した人に確認してほしい
  - 鍵を保持するものでも生パスワードを復元できない
- デメリット
  - 鍵の管理の必要がある

パスワード保存方式の比較

方式	弱パスワードの保護	生パスワード	鍵管理
そのまま保存	不可能	そのまま	不必要
Unix的	stretchで対応	復元不可能	不必要
暗号	可能	復元可能	必要
ハッシュ+暗号	可能	復元不可能	必要
鍵+ハッシュ	可能	復元不可能	必要

個人的には,
鍵の管理が面倒なのでUnix的でよいと考えています.

パスワードの保存 まとめ

- Unix的パスワード保存を解説
- Webシステムでのパスワード保存を考察

パスワードの話題いろいろ

TODO: 以下まだ作りかけ

- 私のパスワード管理法(TODO)
- 攻撃
- 秘密の質問
- 強度
- 定期更新
- マスキング

パスワードに対する攻撃

- 総当たり攻撃
- 辞書攻撃
- ショルダークラック
- キーロガー
- 別のサイトと共通のパスワードを利用しているユーザ
  * 他のサイトでパスワードが漏れて，ログインされる

秘密の質問

- 弊社の例 重要な機能(ポイント交換)を行なう前に秘密の質問を入力させている
  - ユーザがサイトごとに別々のパスワードを付けてくれれば，必要ないのだが...
- よくあるのは小学校の名前とか親の旧姓とか
  - 他者が推測可能なものがある...
- 個人的には第2パスワードとか交換用パスワードなどと呼んで，普通のパスワードと同じように管理してもらうほうがいいのではと考えている

パスワードの強度(1)

文字種を増やすのがよいか，長さを増やすのがよいか?

パスワードの強度(2)

文字種	文字数	総パスワード数
62(英数)	8	218兆
96(英数記号)	8	7213兆
62(英数)	9	13537兆
62(英数)	10	839299兆

- 文字長を伸ばしたほうがいい.
  - 記号を入れることを強制するよりも
    最小の文字長を大きくしたほうがよい.

パスワードの定期更新(1)

パスワードを定期的に更新する意味はあるのか?

パスワードの定期更新(2)

- 通常は意味がない．むしろ有害
  - 攻撃に対する強度はパスワードの強度
    - 定期的に変えても強度はあまり増えない．

      *続パスワードの定期変更は神話なのか － ockeghem(徳丸浩)の日記
    - パスワードの変更により打ち間違えが増え利便性が下がる

パスワードの定期更新(3)

- 意味がある場合
  - 共有アカウントで,
    人員の入れ替えが頻繁にある場合
    - 定期更新によって権限がない人のアクセスを止めれる
    - セキュリティ的には共有アカウントでないほうがよい

パスワードのマスキング

- ショルダークラック vs 利便性
  - 要件に依存する
- 個人的にはユーザが切り替えられるのがいいと思う
  - より個人的には,
    パスワード管理ツールを適当に使うので,
    Webサイトでパスワードを手で入力することがない.
    のでどうでもよい

参考文献

man 3 crypt

Manpage of CRYPT

CRYPTOGRAPHY ENGINEERING

ISBN-13: 978-0470474242

認証技術 パスワードから公開鍵まで

ISBN-13: 978-4274065163