Smart Card HOTP implementation

inexpensive hardware based one time passwords

Agenda

* One Time Passwords
* HOTP Protocol
* Smart Cards and Readers
* PAM
* Open source implementation of hardware based OTP system.

OTP

* One Time Passwords are passwords that can only be used once.
* Generated when needed for each login session or printed out (S/Key) and carried with you.
* S/Key and SecurID are popular implementations.

S/Key

```
0: WAY SKI WHOM LIN HALE TEAM
1: BEST SKIT DOSE SEAM FIR TRIG
2: CREW OLIN GAIN ALP CAW ODE
3: ROVE SHE PUN ACE PAP GOAT
4: ANNA MAIL PUG FISK BUCK BLUM
5: FIR PA ALAN FONT WING KISS
6: HERD RAN YAP NECK TOE FUR
7: DENY SMOG LADY REAR MOOR LOST
8: OMIT NEAT PAD HANG THEN MUTE
9: SHUN INN MAT BEAD EVA MUTT
10: BILE COIL HEEL VET HILT LEAK
```

Text

Why OTP's

* Given the opportunity people will choose easy to guess passwords.
* Password sniffers are a commodity.
* "Use ssh or encrypted VPN's everywhere" isn't practical. Still have a lot of gear that doesn't support ssh or VPN's.

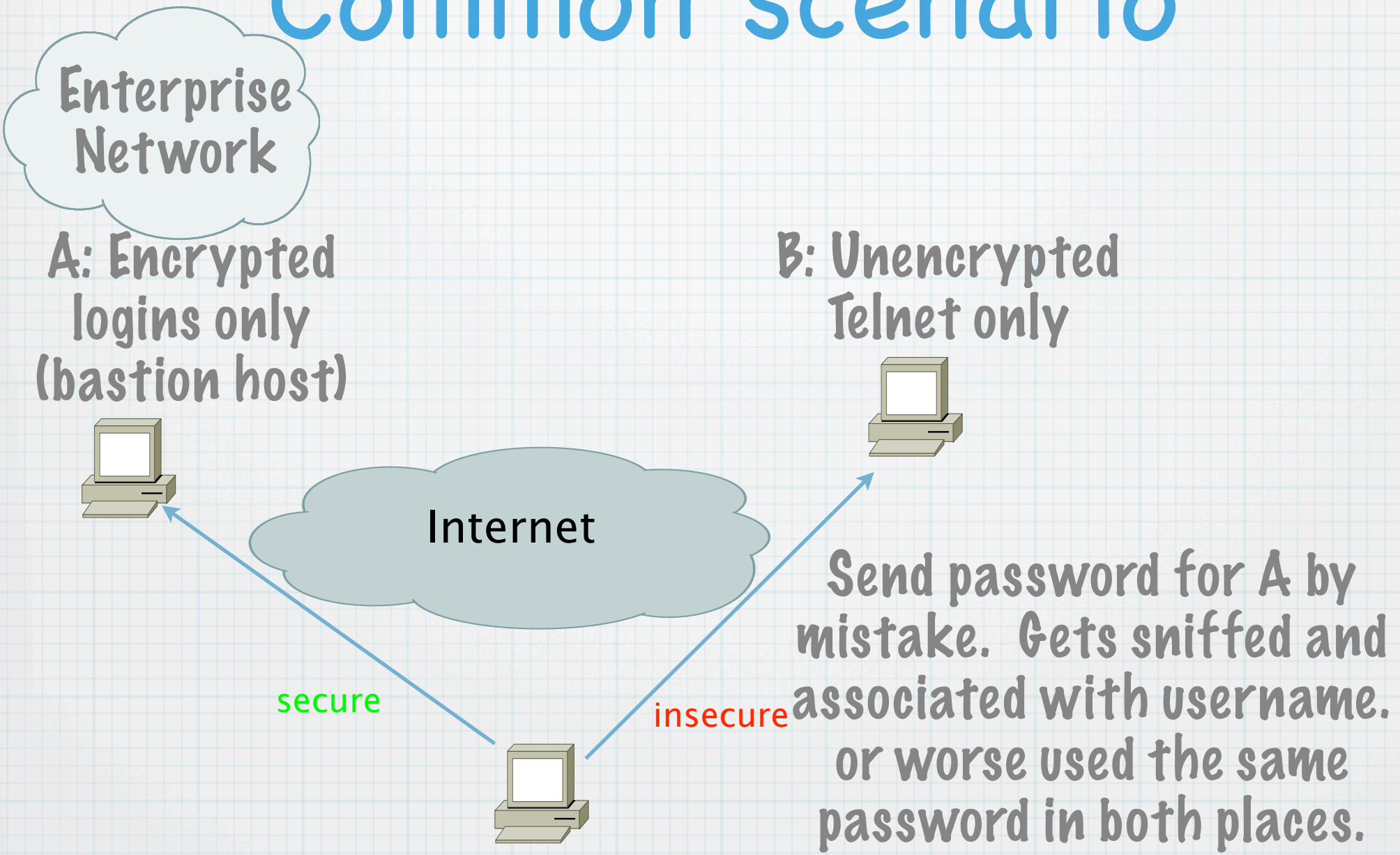
Why OTP's

* Staff tend to get a lot of passwords to remember (POP Mail password, login host, routers switches, terminal servers, encrypted SSH keys, power controllers, Windows logins, etc, etc).
* When one password doesn't work try another.

Why OTP's

* Telnet somehost. Login fails because you typed your "encrypted connections only" bastion host password instead of the one for "somehost" Now your secure password has been sent in the clear.

Common scenario



Why OTP's

* Immune to sniffers - password only works once.
* Immune to bad passwords.
* Two Factor Authentication - "something you have" a device to generate the OTP and "something you know" a PIN or reusable password.

OTP not a cure-all

* Unencrypted/Unauthenticated sessions can still be hijacked, although they can't login again.
* If the device you're logging in from has been r00t3d you lose. Your session can be hijacked after or between authentication.

Barriers to OTP deployment.

* S/Key is free, why doesn't everyone use it. Not for the non tech savvy.
* Secure-ID or other vendors with proprietary solutions. Costs a lot, especially if you're just trying to lock down your personal PC for remote access.

OTP requirements

* Low per user cost and availability in small quantities -- 1.
* Useable by non tech savvy staff.
* Not OS specific, don't want to maintain drivers and/or client for Windows, Mac, Linux, *nix.
* Open Source, Open Standard.

Possible solutions.

- Smart card + reader + HOTP + small library + pam module.
- Smart card and reader issued to users. PAM module runs on login server.

Balance Reader

* Used in Europe for E-cash
* Available in small quantities (1)
* Not very flexible - sends command to smart card asking what to display.

Spyrus PAR II

* PAR II is programmable reader - firmware written to support BasicCard HOTP implementation.

Spyrus PAR II

* A lot more potential since reader is programmable (PIC 16F877).
* Can support multiple keys (login hosts) per card/reader - about 100.
* PIN is required before use.
* Available in small quantities (1)

Spyrus PAR II

* Multiple keys / card : Each system has a key per user. No central server / network requirements.
* User selects hostname from menu to generate HOTP after entering PIN.

Smart Cards

- A few varieties. Contact and Contactless (RFID). May have just EEPROM (memory card) or a microcontroller with EEPROM and RAM.
- Many vendors. Some have a small operating system, some have a large OS (Java based cards).

Smart Cards

* ISO-7816-* defines the physical characteristics, electrical signals and transmission protocols.
* PC/SC - defines low level interface and API.
* What's in the card and running on it are vendor specific.

BasicCard

- ZC 3.9 inexpensive contact type microcontroller based card with 256 bytes of RAM and 8K EEPROM.
- IDE which allows you to program it in a BASIC like language (free)
- Includes Crypto Library.

What's HOTP

* Internet draft detailing an algorithm to generate One Time Passwords based on HMAC.
* HOTP(K,C) = Truncate(HMAC-SHA-1(K,C))

HOTP(K,C) = Truncate (HMAC-SHA-1(K,C))

* K is a secret key, C is a counter which increases with each use, Truncate() transforms the 160 bit result into something that can be typed by hand, HMAC-SHA-1 is a secure message authentication code

HMAC

* Keyed-Hashing for Message Authentication
* Apply a secure hash (MD5, SHA-160, etc) to a Key and Message. Result is a digital signature of the Message which can be validated by parties which know the Key.

HMAC

- A third party can alter the Message but they can't generate a valid HMAC without the Key.
- Alternately for a given Message only parties which know the Key can generate a valid HMAC.

HMAC

- ipad = 0x36 repeated B times (B is the block size of the hash function)
- opad = 0x5C repeated B times
- H(K XOR opad, H(K XOR ipad, text)) where H is the hash function. Will compute the HMAC of (K,text).

HOTP

* Requires a shared Key. One copy is on the Smart Card, one is on the authentication server.
* Requires a loosely synchronized shared Count.

HOTP

- User is presented with a challenge. Uses smart card + reader to display HMAC(K,C). Smart Card increments count which is stored in NV memory.
- Server computes HMAC(K,C) for user.

HOTP

* If the HMAC entered by user matches that of the server then server increments Count and user is authenticated.
* If there is not a match, server repeats adding 1 to the Count up to Window times (window = 10).

HOTP

* If after window tries and there is no match, user is not authenticated.
* Server stores database of {Username,Count,Key,Last,Status}.
* Note, there is not a requirement for a separate authentication server, although it could be implemented this way.

HOTP

* Last is time, used to thwart brute force attacks. Authentication attempts policed to 1 per second. If truncated HMAC is 40 bits then on average will take 17,000 years to brute force an account.
* Status allows disabling of login or HOTP requirement on per user basis.

Getting Started

- Get a ZC3.9 Smart Card, balance reader, and a PC/SC compatible interface. Or just get the BasicCard development kit for about $80.
- Smart Card - $3.10/user
- Balance Reader - $12.35/user or Spyrus PARII ($60/user)
- PC Interface - $10 - $50 (one time)

Getting Started

* Download HOTP software to Smart Card with PC Interface and BCLOAD program (or IDE).
* Add a user to the database with otp-control (also creates a key).

Getting Started

* If using the Spyrus reader download the HOTP firmware. Requires the use of a special RS232 cable available from Spyrus and "PIC downloader 1.08"

Getting Started

- Program the Key(s) into the Smart Card with HOTPT.EXE (to do this requires a second key that's compiled into the HOTPC image.
- Configure PAM to use pam_otp.so

```
sshd    auth    requisite       pam_unix.so     try_first_pass
sshd    auth    required        pam_otp.so      expose_account,display_count
```

Getting Started

* Try to login. Use Smart Card and Balanace reader or Syprus reader to generate challenge response.

```
% ssh 10.0.0.1
Password: <secret>
HOTP Challenge (3): 143abc589a
Last login: Wed Jun  8 20:15:16 2005 from dev1.eng.oar.ne
Copyright (c) 1980, 1983, 1986, 1988, 1990, 1991, 1993, 1994
        The Regents of the University of California.  All rights reserved.

FreeBSD 4.11-STABLE (FIVIR) #1: Sun Mar  6 03:08:47 GMT 2005

Welcome to FreeBSD!
```

Software

* http://www.splintered.net/sw/otp

Deployment

* ~40 user deployment at OARnet with Spyrus PAR2 readers. Max 4 keys/systems per card.
* Balance readers are still useful since they are smaller and fit better on key chain.

Deployment

* Trampoline BSD servers (redundant locations) with HOTP required on all logins only in-band access to OARnet routers, switches, and Unix hosts.
* VPN deployment will follow for other services not easily supported via ssh.

Deployment

* Added OpenVPN authentication module plug-in.
* Added otp-cadmin - Unix command line utility to replace Windows based HOTP Terminal Application. Uses PCSC standard card API.

Summary

* Low startup cost.
* Can be deployed in small quantities (1).
* No requirement for network server - no central point of failure or requirement for operational network.

Summary

- ~ 100 keys (systems) per card when using Spyrus reader.
- SSH support via PAM module.
- OpenVPN support via plug-in authentication module.

Contact

Mark Fullmer

maf@splintered.net

http://www.splintered.net/sw/otp