Unlocking FileVault

An analysis of Apple's disk encryption system

Jacob Appelbaum <jacob@appelbaum.net>
Ralf-Philipp Weinmann <ralf@coderpunks.org>

@23C3 (52°31'14.48"N, 13°24'59.51"E)

2006-12-29

What's this about?

- What is FileVault? Why use FV?
- Practical problems with FileVault
  - Known and unknown attacks against FV
- Reversing on OS X
  - DiskImages Framework
- FileVault crypto details
  - A free implementation
- OSX oddities and more
  - PRNG, swap, sleep images, DMA attacks
- Special guest Hacker Happily Hacking!

Motivation

- General interest in disk cryptography
- Personal data retention
- Protection against theft
- Everyone uses laptops
- Undocumented. Is it secure? How does it work?

The marketing side

- “FileVault secures your home directory by encrypting its entire contents using the Advanced Encryption Standard with 128-bit keys. This high-performance algorithm automatically encrypts and decrypts in real time, so you don’t even know it’s happening.”

... but we do want to know *what's* happening!

- Internals are not (well) documented
  - Exception: man page for `hdiutil(8)`
- DiskImages framework is private (no src, no headers)
  - `/System/Library/PrivateFrameworks/DiskImages.Framework`
- Kernel module not open-sourced

DiskImages framework

- Modular architecture, supports plugins
  - `hdiutil plugins`
- third-party plugin known: VirtualPC disk images
- helpers: `diskimages-helper, hdiejectd`
- `hdiutil(8)`: CLI “front-end”
- `IOHDIXController` kernel module does in-kernel attach and encryption/decryption (shows up as `com.apple.AppleDiskImageController`)

DiskImages framework (2)

- Backing stores:
  - CBSDBackingStore
  - C{RAM,Carbon,Dev,CURL,Vectored}BackingStore
- Encodings
  - **CEncryptedEncoding**
  - C{MacBinary,AppleSingle,UDIF,SegmentedNDIF,SegmentedUDIF,SegmentedUDIFRaw}Encoding
- Shadowed images, compressed images, **sparse images**, message digests on images

Crypto details

- Blocks get encrypted in 4kByte “chunks” AES-128, CBC mode
  - IV := trunc_{128}(HMACSHA1(hmac-key || chunkno))
- Keys are encrypted (“wrapped”) in header of disk image
- Wrapping of keys done using 3DES-EDE
- Two different header formats (v1, v2)
- Version 2 header: support for asymmetrically (RSA) encrypted header

Crypto details / implementation

- Login password used to derive key for unwrapping
  - PBKDF2 (PKCS#5 v2.0), 1000 iterations
- Crypto parts implemented in CDSA/CSSM
  - DiskImages has own AES implementation, pulls in SHA-1 from OpenSSL dylib
- “Apple custom” key wrapping loosely according to RFC 2630 (PKCS#7, section 12.6)
  - in Apple's CDSA provider (open source)

Recovery mechanism

- When enabling FileVault, you can set a master password
- Master password protects FileVault recovery keychain
  - `/Library/Keychains/FileVaultMaster.keychain`
- Recovery keychain contains 1024 bit RSA key
- However, beware:
  1024 bit RSA modulus ≈ 72 bit symmetric key
  (Lenstra-Verheul heuristics)

Headers / versions

- V1 “headers” live at the *end* of the file
- V2 headers live at the beginning
- “Version is the default for non-sparse images. As of OS X 10.4.7, sparse, encrypted images will always use version 2.” (hdiutil man page)
- Meta data at end of the image can lead to “bad” things when compacting.

Password header for version 2

```
uint32_t kdf_algorithm;
uint32_t kdf_prng_algorithm;
uint32_t kdf_iteration_count;
uint32_t kdf_salt_len; /* bytes */
uint8_t  kdf_salt[32];
uint32_t blob_enc_iv_size; /* bytes */
uint8_t  blob_enc_iv[32];
uint32_t blob_enc_key_bits; /* bits */
uint32_t blob_enc_algorithm;
uint32_t blob_enc_padding;
uint32_t blob_enc_mode;
uint32_t encrypted_keyblob_size;
uint8_t  encrypted_keyblob[0x30];
```

Reversing Private Frameworks

- full signatures for C++ code, e.g.:
  - `CEncryptedEncoding::decodePasswordHeader(Security::CssmData const&, CEncryptedEncoding::PasswordHeader const&)`
  - `CEncryptedEncoding::decodePrivateKeyHeader(__CFString const*, CEncryptedEncoding::PrivateKeyHeader const&)`
  - `CEncryptedEncoding::decodeV1Header(Security::CssmData const&, CEncryptedEncoding::V1Header const&)`
  - `CEncryptedEncoding::decrypt(long long, long long, void*)`
- Analysis done using gdb, hdiutil debug output and otool disassemblies
- Would've liked to use the Boomerang reverse compiler...
  - Worked somewhat after a little patching; not used though. Lots of more work to fix it...

Results?

- vfdecrypt
  - Input encrypted dmgs, output decrypted dmgs
  - Works for Version 1 and Version 2 encrypted dmgs
  - Encrypted sparse disk images: only outer layer will be stripped (encryption); still a sparse disk image inside.
  - Very rough code at the moment, but works.
- Cryptographic security depends on more than just AES-128, it's rather
  3DES effective 112bit || AES-128 || RSA-1024

Why we'd like FDE

- Since only `$HOME` is encrypted, all other data is still unprotected.
- Think `/tmp`, log files: `/var/log`, `/System/Logs`
- We'd like to have full disk encryption
- Possible with DiskImages framework (`CDevBackingStore`), but possibly `hdiutil` is not sufficient for setting it up.

OS X PRNG pecularities

- Uses (modified) Yarrow
- Initial entropy determined from system time
- Security Server (`securityd`) feeds entropy to kernel by *writing* to `/dev/random`
  - This data is pulled from the kernel using a debug interface (`KERN_KDGETENTROPY`) every 15 secs.
- Reseeds are very short (50µsec). Predictability of reseed operations.

Attack vectors?

- Found in xnu-792.13.8 and earlier:

```
/*
 * Encryption data.
 * "iv" is the "initial vector".  Ideally, we want to
 * have a different one for each page we encrypt, so that
 * crackers can't find encryption patterns too easily.
 */

[...]

/*
 * No need for locking to protect swap_crypt_ctx_initialized
 * because the first use of encryption will come from the
 * pageout thread (we won't pagein before there's been a pageout)
 * and there's only one pageout thread.
 */

[...]

#define SWAP_CRYPT_AES_KEY_SIZE 128 /* XXX 192 and 256 don't work ! */
```

Firewire

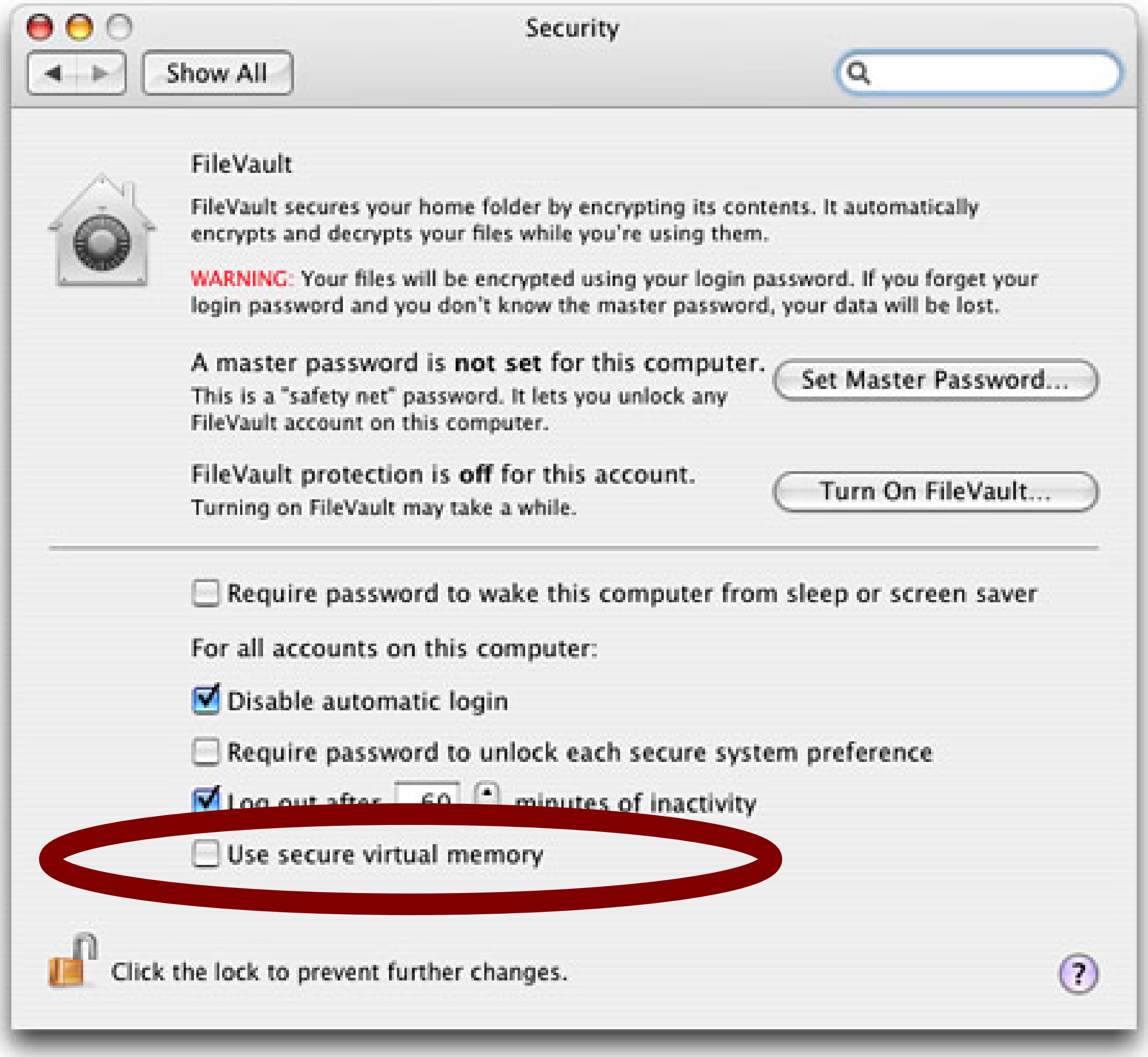
- DMA firewire attacks allow for reading and writing of all system memory
- Possible to own people with an iPod
- Possible to defend against with OpenFirmware or with a patched kernel (see references)
  - Platform dependent

Swap files and memory issues

- Well known issues
- Passwords are not properly scrubbed
- Encrypted swap not on by default in Tiger or even available Panther or below
  - /var/vm/swapfile{0,1,n} containing unhashed user passwords and other sensitive info
- Any ring 0 code can take your keys (remote airport key harvesting anyone?)

But surely everyone knows about encrypted swap?

- **(http://www.apple.com/macosx/features/filevault)**



FileVault

FileVault secures your home folder by encrypting its contents. It automatically encrypts and decrypts your files while you're using them.

WARNING: Your files will be encrypted using your login password. If you forget your login password and you don't know the master password, your data will be lost.

A master password is **not set** for this computer.
This is a "safety net" password. It lets you unlock any FileVault account on this computer.

FileVault protection is **off** for this account.
Turning on FileVault may take a while.

□ Require password to wake this computer from sleep or screen saver

For all accounts on this computer:

☑ Disable automatic login

□ Require password to unlock each secure system preference

☑ Log out after 60 minutes of inactivity

□ Use secure virtual memory

Click the lock to prevent further changes.

Safe Sleep

- Safe sleep is invoked when power runs critically low
- Memory contents written to `/var/vm/sleepimage`
- Safe sleep is careful but not careful enough...
- If encrypted swap is on:
  - contents of the sleep image will be encrypted, but key will be written out in the header (xnu-792.13.8)

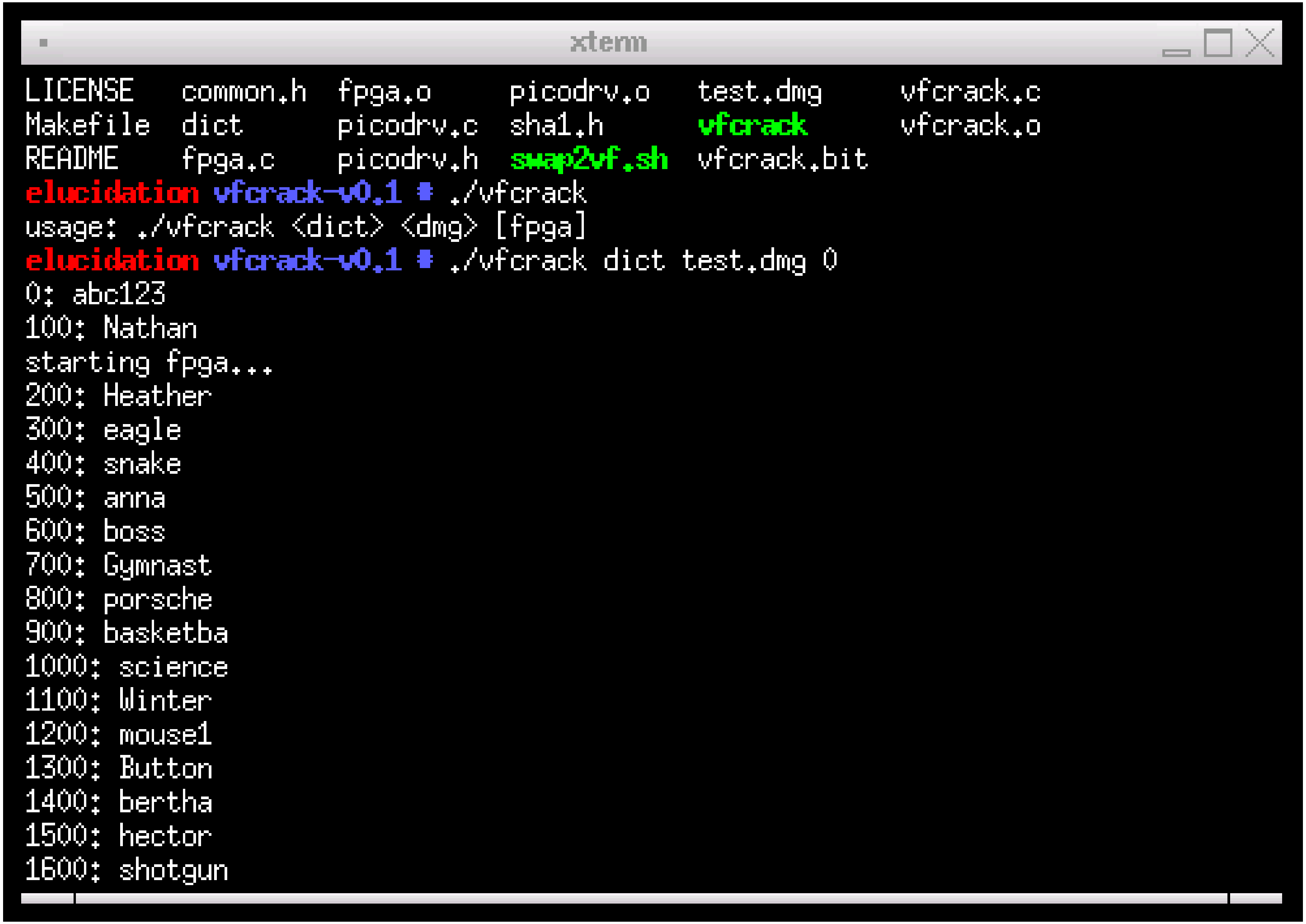
Weak passwords

- Brute force dictionary attacks are possible
- We can typically get around ~200 keys/sec
  - AMD Sempr0n 3300

Special guest appearance

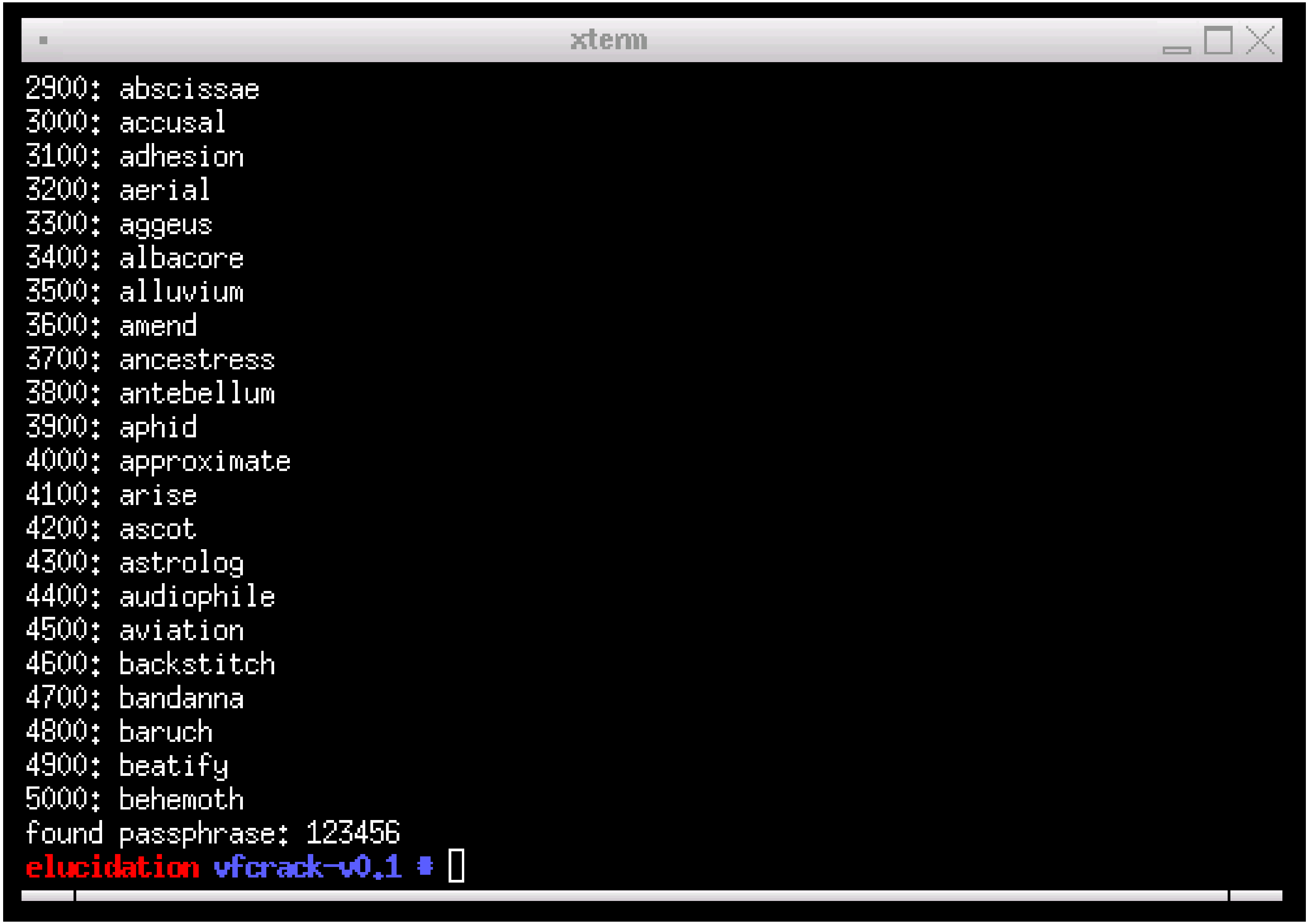
- Please welcome David Hulton
  - Demoing vfcrack

vfcrack working



```
LICENSE   common.h  fpga.o     picodrv.o  test.dmg     vfcrack.c
Makefile  dict      picodrv.c  sha1.h     vfcrack      vfcrack.o
README    fpga.c    picodrv.h  swap2vf.sh vfcrack.bit
elucidation vfcrack-v0.1 # ./vfcrack
usage: ./vfcrack <dict> <dmg> [fpga]
elucidation vfcrack-v0.1 # ./vfcrack dict test.dmg 0
0: abc123
100: Nathan
starting fpga...
200: Heather
300: eagle
400: snake
500: anna
600: boss
700: Gymnast
800: porsche
900: basketba
1000: science
1100: Winter
1200: mouse1
1300: Button
1400: bertha
1500: hector
1600: shotgun
```

vfcrack done



```
2900: abscissae
3000: accusal
3100: adhesion
3200: aerial
3300: aggeus
3400: albacore
3500: alluvium
3600: amend
3700: ancestress
3800: antebellum
3900: aphid
4000: approximate
4100: arise
4200: ascot
4300: astrolog
4400: audiophile
4500: aviation
4600: backstitch
4700: bandanna
4800: baruch
4900: beatify
5000: behemoth
found passphrase: 123456
elucidation vfcrack-v0.1 #
```

vfcrack

- We can typically get around ~200 keys/sec with a normal laptop
- Using a compact flash sized FPGA from pico computing we can increase this dramatically
- We can achieve ~2000+ keys/sec (10x!)
- Demo!

Other fine references

- Firewire DMA attacks - "All your memory are belong to us" @ http://md.hudora.de/presentations/ by Maximillian Dornseif
- Secure your Mac workshop by Angelo Laub @ http://metalab.at/wiki/SYMWorkshop
- DmgBrute by ? - http://fsbsoftware.com/data/dmgBrute.c
- MDE@22c3 by Jacob Appelbaum - http://events.ccc.de/congress/2005/fahrplan/attachments/714-Slides-Modern_Disk_Encryption_Systems.pdf

Code and slide release

- Free and Open Source software
- Cracking with optional FPGA acceleration (Thanks to h1kari) included as **vfcrack**
- Download now from: http://crypto.nsa.org/vilefault/

Thanks!

- Christan Fromme
- David Hulton
- Angelo Laub
- Jennifer Granick
- Luis Miras
- To every person at the EFF
- All the great people in and at the CCC

And thanks most of all...

- Club Mate!

Questions?