Playing Defense with PBKDF2: hashcat and the 1Password Agile Keychain from the defender's point of view

A (rough) day in the life of a defender

Jeffrey Goldberg

Chief Defender of the Dark Arts
AgileBits Inc.
jeff@agilebits.com

Passwords13

Outline

agile bits

AgileBits and 1Password

- 1Password is the flagship product of AgileBits, Inc.
- In 2005 Dave Teare and Roustem Karimov said, "*Wouldn't it be cool if we could use the OS X Keychain to build a form filler and password manager for ourselves?*"
- 1Password is now extremely popular and runs on OS X, iOS, Windows (and there is an Android reader).

Relevant Design Principles

- Sometimes people's encrypted data will get stolen.
  (So we started using PBKDF2 before it was cool)
- We are willing to say "no" to feature requests
  (E.g., we don't have an option for a cascade of AES-Twofish-Serpent)
- Not open-source, but open about security design
  (Most relevant for this audience, we've provided samples and details to those building crackers.)
- Demonstrate that "usable security" is not an oxymoron. We want non-experts to have top-notch security.

Relevant Design Principles

- Sometimes people's encrypted data will get stolen. (So we started using PBKDF2 before it was cool)
- We are willing to say "no" to feature requests (E.g., we don't have an option for a cascade of AES-Twofish-Serpent)
- Not open-source, but open about security design (Most relevant for this audience, we've provided samples and details to those building crackers.)
- Demonstrate that "usable security" is not an oxymoron. We want non-experts to have top-notch security.

agile bits

Relevant Design Principles

- Sometimes people's encrypted data will get stolen.
  (So we started using PBKDF2 before it was cool)
- We are willing to say "no" to feature requests
  (E.g., we don't have an option for a cascade of AES-Twofish-Serpent)
- Not open-source, but open about security design
  (Most relevant for this audience, we've provided samples and details to those building crackers.)
- Demonstrate that "usable security" is not an oxymoron. We want non-experts to have top-notch security.

agile bits

Relevant Design Principles

- Sometimes people's encrypted data will get stolen. (So we started using PBKDF2 before it was cool)
- We are willing to say "no" to feature requests (E.g., we don't have an option for a cascade of AES-Twofish-Serpent)
- Not open-source, but open about security design (Most relevant for this audience, we've provided samples and details to those building crackers.)
- Demonstrate that "usable security" is not an oxymoron. We want non-experts to have top-notch security.

agile bits

Relevant Design Principles

- Sometimes people's encrypted data will get stolen. (So we started using PBKDF2 before it was cool)
- We are willing to say "no" to feature requests (E.g., we don't have an option for a cascade of AES-Twofish-Serpent)
- Not open-source, but open about security design (Most relevant for this audience, we've provided samples and details to those building crackers.)
- Demonstrate that "usable security" is not an oxymoron. We want non-experts to have top-notch security.

agile bits

Relevant Design Principles

- Sometimes people's encrypted data will get stolen.
  (So we started using PBKDF2 before it was cool)
- We are willing to say "no" to feature requests
  (E.g., we don't have an option for a cascade of AES-Twofish-Serpent)
- Not open-source, but open about security design
  (Most relevant for this audience, we've provided samples and details to those building crackers.)
- Demonstrate that "usable security" is not an oxymoron. We want non-experts to have top-notch security.

agile bits

Relevant Design Principles

- Sometimes people's encrypted data will get stolen.
  (So we started using PBKDF2 before it was cool)
- We are willing to say "no" to feature requests
  (E.g., we don't have an option for a cascade of AES-Twofish-Serpent)
- Not open-source, but open about security design
  (Most relevant for this audience, we've provided samples and details to those building crackers.)
- Demonstrate that "usable security" is not an oxymoron. We want non-experts to have top-notch security.

Relevant Design Principles

- Sometimes people's encrypted data will get stolen.
  (So we started using PBKDF2 before it was cool)
- We are willing to say "no" to feature requests
  (E.g., we don't have an option for a cascade of AES-Twofish-Serpent)
- Not open-source, but open about security design
  (Most relevant for this audience, we've provided samples and details to those building crackers.)
- Demonstrate that "usable security" is not an oxymoron. We want non-experts to have top-notch security.

My history with AgileBits

- I'd tried a number of password managers, including rolling my own.
- Once I'd made the switch from Linux on the Desktop to OS X, I was willing to consider a Mac-only solution.
- Examined a number of options. After peppering Dave and Roustem with questions, eventually settled on 1Passwd.
- As a customer, I'd been such an enthusiastic advocate and "explainer" of its security that I was offered a job.

I may be the one who explains crypto and security to the public, but our developers are way smarter than I am.

My history with AgileBits

- I'd tried a number of password managers, including rolling my own.
- Once I'd made the switch from Linux on the Desktop to OS X, I was willing to consider a Mac-only solution.
- Examined a number of options. After peppering Dave and Roustem with questions, eventually settled on 1Passwd.
- As a customer, I'd been such an enthusiastic advocate and "explainer" of its security that I was offered a job.

I may be the one who explains crypto and security to the public, but our developers are way smarter than I am.

1Password data formats

- Original version used the OS X keychain.
- In 2007–2009 transitioned to Agile Keychain Format
- Currently transitioning to 1Password 4 Cloud Keychain Format

For this talk, what is relevant is key derivation in the Agile Keychain Format.

Key derivation in Agile Keychain Format

1. Master Password taken as UTF8 string is handed to PBKDF2-HMAC-SHA1 to derive 256-bits
2. 256-bit split into 128 bit AES key and 128 bit IV for AES-CBC
3. Derived AES key is used to decrypt a 1 kilobyte chunk of data encrypted with AES-CBC with PKCS#7 padding.
4. (Other steps that are not relevant for password cracking.)

Key derivation in Agile Keychain Format

1. Master Password taken as UTF8 string is handed to PBKDF2-HMAC-SHA1 to derive 256-bits
2. 256-bit split into 128 bit AES key and 128 bit IV for AES-CBC
3. Derived AES key is used to decrypt a 1 kilobyte chunk of data encrypted with AES-CBC with PKCS#7 padding.
4. (Other steps that are not relevant for password cracking.)

Key derivation in Agile Keychain Format

1. Master Password taken as UTF8 string is handed to PBKDF2-HMAC-SHA1 to derive 256-bits
2. 256-bit split into 128 bit AES key and 128 bit IV for AES-CBC
3. Derived AES key is used to decrypt a 1 kilobyte chunk of data encrypted with AES-CBC with PKCS#7 padding.
4. (Other steps that are not relevant for password cracking.)

Key derivation in Agile Keychain Format

1. Master Password taken as UTF8 string is handed to PBKDF2-HMAC-SHA1 to derive 256-bits
2. 256-bit split into 128 bit AES key and 128 bit IV for AES-CBC
3. Derived AES key is used to decrypt a 1 kilobyte chunk of data encrypted with AES-CBC with PKCS#7 padding.
4. (Other steps that are not relevant for password cracking.)

1Password flaw and 3M guesses per second

hashcat
@hashcat

Following

Support added to crack 1Password to oclHashcat-plus, 100% computed on GPU! Plus I found an exploitable design flaw hashcat.net/forum/thread-2...

agile bits

Jens' post

Jens Steube's (atom's) post was extremely clear and complete.
Speed up due to:

1. HMAC unpacking in PBKDF2
2. AES key check only needed last two blocks of CBC padded data
3. PBKDF2 weirdness (design flaw)
4. Three million guesses per second.

Jens' post

Jens Steube's (atom's) post was extremely clear and complete.
Speed up due to:

1. HMAC unpacking in PBKDF2
2. AES key check only needed last two blocks of CBC padded data
3. PBKDF2 weirdness (design flaw)
4. Three million guesses per second.

Jens' post

Jens Steube's (atom's) post was extremely clear and complete.
Speed up due to:

1. HMAC unpacking in PBKDF2
2. AES key check only needed last two blocks of CBC padded data
3. PBKDF2 weirdness (design flaw)
4. Three million guesses per second.

Jens' post

Jens Steube's (atom's) post was extremely clear and complete.
Speed up due to:

1. HMAC unpacking in PBKDF2
2. AES key check only needed last two blocks of CBC padded data
3. PBKDF2 weirdness (design flaw)
4. Three million guesses per second.

Jens' post

Jens Steube's (atom's) post was extremely clear and complete.
Speed up due to:

1. HMAC unpacking in PBKDF2
2. AES key check only needed last two blocks of CBC padded data
3. PBKDF2 weirdness (design flaw)
4. Three million guesses per second.

Decomposing PBKDF2

- PBKDF2 is composed of HMAC calls
- HMAC is composed of hash function (SHA1) calls.
- SHA1 is composed of compression function calls.

Half of the compressions are the same for each PBKDF2 round, so decomposing can save half of those compressions.

AES-CBC padding

To verify a candidate key,

1. Take the penultimate block as an IV (The real *initial* IV isn't needed)
2. Decrypt the last block
3. Verify padding

So

- verifying a key can be done very quickly
- verifying a key doesn't require the real initial IV

AES-CBC padding

To verify a candidate key,

1. Take the penultimate block as an IV (The real *initial* IV isn't needed)
2. Decrypt the last block
3. Verify padding

So

- verifying a key can be done very quickly
- verifying a key doesn't require the real initial IV

AES-CBC padding

To verify a candidate key,

1. Take the penultimate block as an IV (The real *initial* IV isn't needed)
2. Decrypt the last block
3. Verify padding

So

- verifying a key can be done very quickly
- verifying a key doesn't require the real initial IV

Calling PBKDF2 twice?

Under peculiar circumstances, a defender must perform twice as many HMACs as the attacker.

- The requested number of bits is more than the output of the hash
- The attacker only needs the first portion of the PBKDF2 output.

Attacker has to run only half the compressions as the defender does

Double the defender efforts

1Password for the Agile Keychain Format,

- Asked for 256 bits of output from PBKDF2-HMAC-SHA1 (SHA1 natively produces 160 bits of output.)
- Used the first 128 bits for an AES key. (last 128 bits were for the IV).

The Defender needs to run through twice as many computations as attacker if attacker doesn't need the IV.

3 Million Guesses per Second

Performing all of the operations on GPUs, along with all of these other optimizations, allowed this configuration to try approximate 3 million guesses per second.

(No doubt y'all have achieved faster rates by now on beefed up hardware.)

3 Million Guesses per Second

Performing all of the operations on GPUs, along with all of these other optimizations, allowed this configuration to try approximate 3 million guesses per second.
(No doubt y'all have achieved faster rates by now on beefed up hardware.)

My 1st Response

My first response on the hashcat forum was . . .

- Quick
- Extremely well received (at least on HN and on Twitter)
- Entirely missed the point

My 1st Response

My first response on the hashcat forum was . . .

- Quick
- Extremely well received (at least on HN and on Twitter)
- Entirely missed the point

My 1st Response

My first response on the hashcat forum was . . .

- Quick
- Extremely well received (at least on HN and on Twitter)
- Entirely missed the point

My 1st Response

My first response on the hashcat forum was . . .

- Quick
- Extremely well received (at least on HN and on Twitter)
- Entirely missed the point

A Little Help from My Friends

Throughout the very long day, I did come to understand.

Community help I received enormous help and support from the community at large. It would be hard to overstate this point.

I'd love to acknowledge everyone here, but the list would be too long.

AgileBits support AgileBits staff who handled the flood of queries gave me the ability to focus on developing a better understanding.

Eventual CBC Padding Response

The trick used by hashcat (and others before them) is a clever trick, but . . .

- It's the job of PBKDF2 to slow down cracking. Not of AES.
- The big problem with CBC padding is in Chosen Ciphertext attacks, but that is not what is being exploited here *pace* my initial irrelevant response about authenticated encryption.
- The 1Password 4 Cloud Keychain Format uses a padding scheme that doesn't allow this.

Without this trick, attacker would have needed real IV and would not have been able to exploit the PBKDF2 flaw.

agile bits

Eventual CBC Padding Response

The trick used by hashcat (and others before them) is a clever trick, but . . .

- It's the job of PBKDF2 to slow down cracking. Not of AES.
- The big problem with CBC padding is in Chosen Ciphertext attacks, but that is not what is being exploited here
  pace my initial irrelevant response about authenticated encryption.
- The 1Password 4 Cloud Keychain Format uses a padding scheme that doesn't allow this.

Without this trick, attacker would have needed real IV and would not have been able to exploit the PBKDF2 flaw.

agile bits

Eventual CBC Padding Response

The trick used by hashcat (and others before them) is a clever trick, but . . .

- It's the job of PBKDF2 to slow down cracking. Not of AES.
- The big problem with CBC padding is in Chosen Ciphertext attacks, but that is not what is being exploited here *pace* my initial irrelevant response about authenticated encryption.
- The 1Password 4 Cloud Keychain Format uses a padding scheme that doesn't allow this.

Without this trick, attacker would have needed real IV and would not have been able to exploit the PBKDF2 flaw.

agile bits

Eventual CBC Padding Response

The trick used by hashcat (and others before them) is a clever trick, but . . .

- It's the job of PBKDF2 to slow down cracking. Not of AES.
- The big problem with CBC padding is in Chosen Ciphertext attacks, but that is not what is being exploited here *pace* my initial irrelevant response about authenticated encryption.
- The 1Password 4 Cloud Keychain Format uses a padding scheme that doesn't allow this.

Without this trick, attacker would have needed real IV and would not have been able to exploit the PBKDF2 flaw.

agile bits

Eventual CBC Padding Response

The trick used by hashcat (and others before them) is a clever trick, but . . .

- It's the job of PBKDF2 to slow down cracking. Not of AES.
- The big problem with CBC padding is in Chosen Ciphertext attacks, but that is not what is being exploited here *pace* my initial irrelevant response about authenticated encryption.
- The 1Password 4 Cloud Keychain Format uses a padding scheme that doesn't allow this.

Without this trick, attacker would have needed real IV and would not have been able to exploit the PBKDF2 flaw.

agile bits

HMAC Unpacking Response

- The optimization of PBKDF2 is as available to defenders as it is to attackers.
- It is a clever optimization, but it isn't new. JtR has been using it for some time, and . . .
- The OpenSSL implementation of PBKDF2 does not make that optimization, but it looks like almost everyone else does.
  - E.g., confirmed that Peter Gutman's `cryptlib` uses it.
  - Under NDA with Apple, but we are happy to continue using Apple's CommonCrypto.

Difficult Responses

The previous ones were the "easy" responses, once we fully understood Jens' achievement. But there are more difficult ones that remain.

- Were we doing something wrong in calling PBKDF2 as we did?
- What can we do (or tell our users) about *3 million guesses per second*?

Key sets from PBKDF2

[O]ne might derive a set of keys with a single application of a key derivation function, rather than derive each key with a separate application of the function. The keys in the set would be obtained as substrings of the output of the key derivation function. [RFC 2898, §3]

More Bits than Hash

PBKDF1	*The length of the derived key [by PBKDF1] is bounded by the length of the hash function output [RFC 2898, §5.1]*
PBKDF2	*The length of the derived key [by PBKDF2] is essentially unbounded. [RFC 2898, §5.2]*

It's Not Whether You Win or Lose. . .

. . . But How You Lay the Blame

- 1Password used PBKDF2 according the specification.
- We still got bitten by what turns out to be a design flaw in PBKDF2.

Missing the Old Days

- When we first wrote about PBKDF2 for our customers, we told them that, "*PBKDF2 reduces what would be millions of guesses per second to just thousands of guess per second.*"
- As defenders, we need to acknowledge and address the kinds of cracking speeds that have and will be achieved.
- By "we", I mean everyone who has to protect high-value, human-usable, passwords.
  (For passwords that humans don't need to remember or type, just use a password manager!)

Missing the Old Days

- When we first wrote about PBKDF2 for our customers, we told them that, "*PBKDF2 reduces what would be millions of guesses per second to just thousands of guess per second.*"
- As defenders, we need to acknowledge and address the kinds of cracking speeds that have and will be achieved.
- By "we", I mean everyone who has to protect high-value, human-usable, passwords.
  (For passwords that humans don't need to remember or type, just use a password manager!)

Increasing PBKDF2 iterations

- The sample Jens tested used 1000 iterations.
- For years, 1Password has been creating new data files $\geq$ 10000 iterations.
- Now all versions of 1Password will move legacy data from 1000 iterations to $\geq$ 10000 on password change

Limits to Iterations

- Strength rises only linearly with iterations.
- Data files must be usable on small, portable devices.
  If a Mac Pro can run 100,000 iterations without annoying user we need still must unlock the same data on a phone.
  - Too slow on older phones.
  - Consumes too much battery on portable devices.

Demand For More Iterations

- Customers request more PBKDF2 iterations
- Customers request configurable numbers of iterations.
- We say, "No". (See, I told you those Design Principles would be relevant.)
  We try to remind people that adding a single randomly chosen digit to the end of their password adds as much strength as going from going from 20,000 to 200,000 iterations.

Demand For More Iterations

- Customers request more PBKDF2 iterations
- Customers request configurable numbers of iterations.
- We say, "No". (See, I told you those Design Principles would be relevant.)
  We try to remind people that adding a single randomly chosen digit to the end of their password adds as much strength as going from going from 20,000 to 200,000 iterations.

Demand For More Iterations

- Customers request more PBKDF2 iterations
- Customers request configurable numbers of iterations.
- We say, "No". (See, I told you those Design Principles would be relevant.)

  We try to remind people that adding a single randomly chosen digit to the end of their password adds as much strength as going from going from 20,000 to 200,000 iterations.

Demand For More Iterations

- Customers request more PBKDF2 iterations
- Customers request configurable numbers of iterations.
- We say, "No". (See, I told you those Design Principles would be relevant.)
  We try to remind people that adding a single randomly chosen digit to the end of their password adds as much strength as going from going from 20,000 to 200,000 iterations.

Waiting for PBKDF2 Successor

Solution Alternative to PBKDF2

Problems

- Would prefer KDFs from standard libraries.
- Are uncertain about memory parameters on portable devices
- Want to see more review of a PBKDF2 successor

So, we are waiting for results of the Password Hashing Competition.

Encourage Better Master Passwords

Solution Get people to use better Master Passwords

Problems Passwords for password managements systems . . .

- need to be memorable by humans
- need to be typed on portable devices

Protect data from capture

Solution You can't crack the data if you never get it.

Problems

- Data stored on personal computers and devices is vulnerable to capture
- Data stored on synchronization services are vulnerable to capture

Offer Key splitting option

Solution Decryption key is XOR of password-derived key and something else

- Stored on some "secure" dongle.
- Stored in local (never synchronized) file

Problems

- Needs to work on every platform we support.
- Potential for catastrophic data loss greatly increased

  No matter what warnings we put on enabling such a feature, people will opt for "more secure" even if it isn't the right option for them.

Offer Key splitting option

Solution Decryption key is XOR of password-derived key and something else

- Stored on some "secure" dongle.
- Stored in local (never synchronized) file

Problems

- Needs to work on every platform we support.
- Potential for catastrophic data loss greatly increased

  No matter what warnings we put on enabling such a feature, people will opt for "more secure" even if it isn't the right option for them.

agile bits

Offer Key splitting option

Solution Decryption key is XOR of password-derived key and something else

- Stored on some "secure" dongle.
- Stored in local (never synchronized) file

Problems

- Needs to work on every platform we support.
- Potential for catastrophic data loss greatly increased

  No matter what warnings we put on enabling such a feature, people will opt for "more secure" even if it isn't the right option for them.

Offer Key splitting option

Solution Decryption key is XOR of password-derived key and something else

- Stored on some "secure" dongle.
- Stored in local (never synchronized) file

Problems

- Needs to work on every platform we support.
- Potential for catastrophic data loss greatly increased

  No matter what warnings we put on enabling such a feature, people will opt for "more secure" even if it isn't the right option for them.

Greater Resistance in New Format

The 1Password 4 Cloud Keychain Format . . .

- . . . uses HMAC-SHA512 to get 512 bits of data (so avoids the PBKDF2 design flaw)
- . . . uses HMAC-SHA512, which should be substantially harder to process on GPUs than SHA1.
- . . . does not use PKCS#7 padding
- . . . has much higher minimum hardware requirements than the Agile Keychain Format, and so can use higher PBKDF2 iterations.
- Format now contains version information, so easier path to dropping in a successor to PBKDF2 in future.

agile bits

Greater Resistance in New Format

The 1Password 4 Cloud Keychain Format . . .

- . . . uses HMAC-SHA512 to get 512 bits of data (so avoids the PBKDF2 design flaw)
- . . . uses HMAC-SHA512, which should be substantially harder to process on GPUs than SHA1.
- . . . does not use PKCS#7 padding
- . . . has much higher minimum hardware requirements than the Agile Keychain Format, and so can use higher PBKDF2 iterations.
- Format now contains version information, so easier path to dropping in a successor to PBKDF2 in future.

agile bits

Greater Resistance in New Format

The 1Password 4 Cloud Keychain Format . . .

- . . . uses HMAC-SHA512 to get 512 bits of data (so avoids the PBKDF2 design flaw)
- . . . uses HMAC-SHA512, which should be substantially harder to process on GPUs than SHA1.
- . . . does not use PKCS#7 padding
- . . . has much higher minimum hardware requirements than the Agile Keychain Format, and so can use higher PBKDF2 iterations.
- Format now contains version information, so easier path to dropping in a successor to PBKDF2 in future.

agile bits

Greater Resistance in New Format

The 1Password 4 Cloud Keychain Format . . .

- . . . uses HMAC-SHA512 to get 512 bits of data (so avoids the PBKDF2 design flaw)
- . . . uses HMAC-SHA512, which should be substantially harder to process on GPUs than SHA1.
- . . . does not use PKCS#7 padding
- . . . has much higher minimum hardware requirements than the Agile Keychain Format, and so can use higher PBKDF2 iterations.
- Format now contains version information, so easier path to dropping in a successor to PBKDF2 in future.

agile bits

Greater Resistance in New Format

The 1Password 4 Cloud Keychain Format . . .

- . . . uses HMAC-SHA512 to get 512 bits of data (so avoids the PBKDF2 design flaw)
- . . . uses HMAC-SHA512, which should be substantially harder to process on GPUs than SHA1.
- . . . does not use PKCS#7 padding
- . . . has much higher minimum hardware requirements than the Agile Keychain Format, and so can use higher PBKDF2 iterations.
- Format now contains version information, so easier path to dropping in a successor to PBKDF2 in future.

agile bits

Greater Resistance in New Format

The 1Password 4 Cloud Keychain Format . . .

- . . . uses HMAC-SHA512 to get 512 bits of data (so avoids the PBKDF2 design flaw)
- . . . uses HMAC-SHA512, which should be substantially harder to process on GPUs than SHA1.
- . . . does not use PKCS#7 padding
- . . . has much higher minimum hardware requirements than the Agile Keychain Format, and so can use higher PBKDF2 iterations.
- Format now contains version information, so easier path to dropping in a successor to PBKDF2 in future.

agile bits

Helping with better Master Passwords

- We can provide tools or advice
  - . . . to encourage better Master Password choice
  - . . . to enable people to chose better Master Passwords
- We will continue to publicize the results of crackers as a way to remind 1Password users that their security depends on the strength of their Master Passwords.
  What you — crackers working in the open — do helps us keep our customers more secure.

Helping with better Master Passwords

- We can provide tools or advice
  - . . . to encourage better Master Password choice
  - . . . to enable people to chose better Master Passwords
- We will continue to publicize the results of crackers as a way to remind 1Password users that their security depends on the strength of their Master Passwords.
  What you — crackers working in the open — do helps us keep our customers more secure.