More Cheating
More Cheating
Dirty COW Vulnerability

- Race condition involving memory mapped files which allows user processes to write to root-owned files
Dirty COW Fixed

commit 19be0eaffa3ac7d8eb6784ad9bdbc7d67ed8e619
Author: Linus Torvalds torvalds@linux-foundation.org
Date: Thu Oct 13 20:07:36 2016 GMT

This is an ancient bug that was actually attempted to be fixed once (badly) by me eleven years ago in commit 4ceb5db9757a ("Fix get_user_pages() race for write access") but that was then undone due to problems on s390 by commit f33ea7f404e5 ("fix get_user_pages bug").
Dirty COW Vulnerability

• `madvise(map,100,MADV_DONTNEED)`
• `write("/proc/self/mem")`

• Eventually writes to a file in the middle of page table updates, causing inappropriate file overwriting.
Recap: Block Ciphers

- Operates on a single chunk ("block") of plaintext
  - For example, 64 bits for DES, 128 bits for AES
  - Each key defines a different permutation
  - Same key is reused for each block (can use short keys)
Electronic Code Book (ECB) Mode

- Don’t use ECB mode
Cipher Block Chaining (CBC) Mode: Encryption

 Initialization vector (random)

Sent with ciphertext

plaintext

\[ \text{Initialization vector (random)} \]

\[ + \]

\[ \text{key} \]

\[ + \]

\[ \text{key} \]

\[ + \]

\[ \text{key} \]

\[ + \]

\[ \text{key} \]

block cipher

block cipher

block cipher

block cipher

Sent with ciphertext
CBC Mode: Decryption

Initialization vector → plaintext

Key → decrypt

Key → decrypt

Key → decrypt

Key → decrypt

ciphertext
ECB vs. CBC

AES in ECB mode

AES in CBC mode

Similar plaintext blocks produce similar ciphertext blocks (not good!)

[Picture due to Bart Preneel]
Counter Mode (CTR): Encryption

Initial ctr (random) → ctr → Key → block cipher → pt → ciphertext

ctr+1 → Key → block cipher → pt → ciphertext

ctr+2 → Key → block cipher → pt → ciphertext

ctr+3 → Key → block cipher → pt → ciphertext
Counter Mode (CTR): Decryption

Initial $\text{ctr}$

$\text{ctr} \rightarrow \text{ctr+1} \rightarrow \text{ctr+2} \rightarrow \text{ctr+3}$

$\oplus$ Key $\oplus$ Key $\oplus$ Key

$\text{ct} \rightarrow \text{ct} \rightarrow \text{ct} \rightarrow \text{ct}$

$\text{block cipher} \rightarrow \text{block cipher} \rightarrow \text{block cipher} \rightarrow \text{block cipher}$

$\rightarrow \rightarrow \rightarrow \rightarrow$

$\text{pt} \rightarrow \text{pt} \rightarrow \text{pt} \rightarrow \text{pt}$

$\text{K} \rightarrow \text{K} \rightarrow \text{K} \rightarrow \text{K}$
How Can a Cipher Be Attacked?

• Attackers knows ciphertext and encryption algthm
  – What else does the attacker know? Depends on the application in which the cipher is used!

• Ciphertext-only attack

• KPA: Known-plaintext attack (stronger)
  – Knows some plaintext-ciphertext pairs

• CPA: Chosen-plaintext attack (even stronger)
  – Can obtain ciphertext for any plaintext of his choice

• CCA: Chosen-ciphertext attack (very strong)
  – Can decrypt any ciphertext except the target
Ex: Chosen Plaintext Attacks

“Let’s plan an attack on AF”

[wikipedia]
Ex: Chosen Plaintext Attacks

“This is Midway Island, we’re low on supplies”

[wikipedia]
Ex: Chosen Plaintext Attacks

“AF is low on supplies”
Ex: Chosen Plaintext Attack

• When the allies planted mines in the ocean, the German Navy would send messages about those locations to warn their ships.

[wikipedia]
Examples of Chosen Ciphertext Attacks

• Some serious attacks against SSH have been based on Chosen Ciphertext Attacks

• Example: send chosen ciphertext to SSH server, see whether it responds with an error or not.
Examples of Chosen Ciphertext Attacks

• Imagine a system with very few commands, e.g., a military system which responds to the commands (“FIRE”) and (“DON’T FIRE”). Try sending ciphertexts and observe in real life whether the weapon fires or not.

• The side effects of the command serve as a “decryption” of your ciphertext.
Very Informal Intuition

• Security against chosen-plaintext attack (CPA)
  – Ciphertext leaks no information about the plaintext
  – Even if the attacker correctly guesses the plaintext, he cannot verify his guess
  – Every ciphertext is unique, encrypting same message twice produces completely different ciphertexts
Message Authentication Codes
So Far: Achieving Privacy

Encryption schemes: A tool for protecting privacy.

Message = M
Ciphertext = C
Integrity and authentication: only someone who knows KEY can compute correct MAC for a given message.
Reminder: CBC Mode Encryption

Initialization vector (random) → plaintext → block cipher → ciphertext

key → block cipher → ciphertext
CBC-MAC

plaintext

block cipher
+ key
block cipher
+ key
block cipher
+ key
block cipher
+ key

TAG
• Not secure when system may MAC messages of different lengths.
Hash Functions
Application: Password Hashing

• Instead of user password, store $\text{hash(password)}$
• When user enters a password, compute its hash and compare with the entry in the password file
  – System does not store actual passwords!
  – Cannot go from hash to password!
• Why is hashing better than encryption here?
• Does hashing protect weak, easily guessable passwords?
**Application: Software Integrity**

**Goal:** Software manufacturer wants to ensure file is received by users without modification.

**Idea:** given `goodFile` and `hash(goodFile)`, very hard to find `badFile` such that `hash(goodFile)=hash(badFile)`
Hash Functions: Main Idea

• Hash function $H$ is a lossy compression function
  – Collision: $h(x) = h(x')$ for distinct inputs $x, x'$

• $H(x)$ should look “random”
  – Every bit equally likely to be 0 or 1

• Cryptographic hash function needs a few properties...
Property 1: One-Way

• The hash should be hard to invert
  – “Preimage resistance”
  – Let $h(x') = y \in \{0,1\}^n$ for random $x'$
  – Given $y$, it should be hard to find any $x$ such that $h(x) = y$
Security Mindset Anecdote

• A clever example of a one-way function: phone books.
Security Mindset Anecdote

• A clever example of a one-way function: phone books.

• Hash(name) = Phone number of person with that name
Security Mindset Anecdote

• Easy to compute forward (phonebook is alphabetical)

• Hard to invert backward (must search n/2 pages on average to find person by phone number)
Property 2: Collision Resistance

• Should be hard to find $x \neq x'$ such that $h(x) = h(x')$
Birthday Paradox

• Expect birthday “collision” half the time with a room of only 23 people.
  • Approximate: 50% probability = \( \sqrt{365} \).

• Why is this important for cryptography?
  – \( 2^{128} \) different 128-bit values
    • Pick one value at random. To exhaustively search for this value requires trying on average \( 2^{127} \) values.
    • Expect “collision” after selecting approximately \( 2^{64} \) random values.
    • 64 bits of security against collision attacks, not 128 bits.
Property 2: Collision Resistance

• Should be hard to find $x \neq x'$ such that $h(x) = h(x')$
• Birthday paradox (informal)
  – Let $t$ be the number of values $x, x', x'' ...$ we need to look at before finding the first pair $x, x'$ s.t. $h(x) = h(x')$
  – What is probability of collision for each pair $x, x'$? $1/2^n$
  – How many pairs would we need to look at before finding the first collision? $O(2^n)$
  – How many pairs $x, x'$ total? $\text{Choose}(t, 2) = t(t-1)/2 \sim O(t^2)$
  – What is $t$, the number of values we need to look at? $2^{n/2}$
• Brute-force collision search is only $O(2^{n/2})$, not $O(2^n)$
  – For SHA-1, this means $O(2^{80})$ vs. $O(2^{160})$
Property 2: Collision Resistance

• Should be hard to find \( x \neq x' \) such that \( h(x) = h(x') \)

• Birthday paradox means that brute-force collision search is only \( O(2^{n/2}) \), not \( O(2^n) \)
  – For SHA-1, this means \( O(2^{80}) \) vs. \( O(2^{160}) \)
Property 3: Weak Collision Resistance

• Given randomly chosen $x$, hard to find $x'$ such that $h(x)=h(x')$
  – Attacker must find collision for a specific $x$. By contrast, to break collision resistance it is enough to find any collision.
  – Brute-force attack requires $O(2^n)$ time

• Weak collision resistance does not imply collision resistance.
Properties of a Cryptographic Hash Function

• One-wayness
  – Given \( h(x) \)...

• Collision resistance
  – Hard to find...

• Weak collision resistance
  – Hard to find...
Properties of a Cryptographic Hash Function

• One-wayness
  – Given $h(x)$: hard to find $x$

• Collision resistance
  – Hard to find $x \neq x'$ s.t. $h(x) == h(x')$

• Weak collision resistance
  – Hard to find $x \neq x'$ s.t. $h(x) == h(x')$
    for specific, random $x$