Sunday, February 22. 2009

Proof-of-concept hack for encrypted direct messages on Twitter

A lazy afternoon after drinking coffee at my prefered coffee dealer. So i had some time for a proof of concept. Fsck ... i didn´t knew how rusty my Perl knowledge got over the time. I just tried to implement an encrypted twitter client. It´s really a bad,bad,bad hack just to test the concept. The code isn´t cleaned up ... contains many artifacts of abandoned ideas and its highly probable to fall on its nose just by using it for something other as for the proof of concept.

The idea of sending the crypted message is pretty simple. The text is encypted with the public key of the receiver. The script uses the Gnu Privacy Guard for this task.

One important task is the distribition of the public key. I´ve started to implement the code for using the Biography field in the Account settings of Twitter or identi.ca for storing the fingerprint and the URL of the public key for the distribution part. I didn´t implemented the part suck the public key and importing it to pgp so far but that would be straightfoward. My idea is to encode both informations in a way, that a encryption enabled twitter client would be able to gather the key whereever the user stores it and validate it by the fingerprint if it´s really the one the twitter user wants to use for encryption. In the Biography of one of users in my test you will find the following text: ((CT|http://www.c0t0d0s0.org/keys/c0t0d0s0bob.pub|F2F1 2771 8C97 1907 7C4C FC98 BECB F313 A24A 72C4))The actual code assumes that the public key of the the receiver is already in the keyring of the sender system. Actually the receiver and the transmitter were on the same system in my proof of concept. I´ve generated two key pairs for my test:$ gpg --list-keys
[...]
/jmoekamp/.gnupg/pubring.gpg
----
pub   1024D/A24A72C4 2009-02-22
uid   c0t0d0s0bob
sub   1024g/A5A8CFC2 2009-02-22

pub   1024D/BC8D1D5C 2009-02-22
uid   c0t0d0s0alice
sub   2048g/10087FB9 2009-02-22The realname is the Twitter name respectively the identi.ca name of the user.

Well, after encryption the script strips off all non-cyphertext. The GnuPG delivers the cyphertext in a handy linelength, so we don´t have to seperate them ourself. I used the remaining chars per twitter message for some metainformation to ease the reassembly of the cyphertext.

At first a MD5 hash is calculated over the cyphertext. This is the message id of the encrypted message. After this step the cybertext is seperated in lines. Every line is prepended by the cTweet magic to sign it as a crypted tweet. After this, the message id, the line number, and the number of total lines of the cyphertext ist appended. Finally a line of the cyphertext is appended to the message. This message is send out to twitter.

The message is hardcoded in this example. I´ve choosen a rather long text to show, that you can send direct messages longer than 140 characters. As we need message reassembly in all cases (even a short text is up to 4-5 messages long because of it´s encryption). I´ve inserted some linebreaks in the message for better readability. In my script it´s one long line. The sender and receiver are hardcoded as well. In my example c0t0d0s0alice wants to send c0t0d0s0bob a secret message. Both use the Twitter API compatible service identi.ca.Okay, let´s starting with the sender. This is ./sendencryptedtwitter.pl: #!/usr/bin/perl
use Net::Twitter;
use Crypt::GPG;
use Digest::MD5 qw(md5 md5_hex md5_base64);

$to_user = "c0t0d0s0bob";
$user="c0t0d0s0alice";
$password= "r78hpQbKA2vn9cu6Gy3R";
$message = "Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium
doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis
et quasi architecto beatae vitae dicta sunt explicabo. Nemo enim ipsam voluptatem quia

voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt. Neque porro quisquam est, qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit, sed quia non numquam eius modi tempora incidunt ut labore et dolore magnam aliquam quaerat voluptatem.
Ut enim ad minima veniam, quis nostrum exercitationem ullam corporis suscipit laboriosam, nisi ut aliquid ex ea commodi consequatur? Quis autem vel eum iure reprehenderit qui in ea voluptate velit esse quam nihil molestiae consequatur, vel illum qui dolorem eum fugiat quo voluptas nulla pariatur?";

```
$identica = Net::Twitter->new(username=>$user, password=>$password, identica=>1);
$result = $identica->show_user({id => $to_user});
%user = %{$result};
$description=$user{'description'};
($keyurl,$keyfingerprint) = $description =~ /\(\(CT|(.*)|(.*)\)\)/gi;

my $gpg = new Crypt::GPG;
$gpg->gpgbin('/usr/bin/gpg');
my @encrypted = $gpg->encrypt ($message, $to_user);

foreach $i (@encrypted) {
 $rawencrypted=$rawencrypted . $i;
}

$digest = md5_hex($rawencrypted);

@cookedencrypted=split(/\n/,$rawencrypted);

foreach $i (@cookedencrypted) {
 next if $i =~ /----/;
 next if $i =~ /Version:/;
 next if $i =~ /Comment:/;
 next if $i =~ /^$/;
 push(@cleanedencrypted,$i);
}

$cleanedlength=scalar(@cleanedencrypted);

foreach $i (@cleanedencrypted) {
 $counter++;
 $sendline="cTweet:$digest:$counter:$cleanedlength:$i";
 $result = $identica->new_direct_message({user => $to_user, text => $sendline});
}This code results into 18 direct
```

messagescTweet:85f36fe1afa2f27051d2b74a04e75cf7:1:18:hQEOA7Ad3jalqM/CEAP9Hv0LUlMeYiPtzr3EztC/j7A0wIo2F7QhuNB4s9mUj7YX
cTweet:85f36fe1afa2f27051d2b74a04e75cf7:2:18:DmrcfdPzrLeh9psd8O8aF47wvgMfvVllr4KW9srMB8aPVyGwR7AXJmmteD0uYU/+
cTweet:85f36fe1afa2f27051d2b74a04e75cf7:3:18:I7SrJiWOfjmt5wfsdPMBN4PdWX4cvo3kdQP19sZU4fgnkHsbA7UnThirVE445BQE
cTweet:85f36fe1afa2f27051d2b74a04e75cf7:4:18:AKbnc/ybJDi7I+A6CIV33+15LLO9VVzVwZWRITPx/+NQiVZAmI8wnPnqqPzb4Bc0
cTweet:85f36fe1afa2f27051d2b74a04e75cf7:5:18:cVqmxttkkkZFdg7GtC9gI6Mg7Esi0ubysvQtQbQ71OGB+DgoJG/Od578I+O7yyiw
cTweet:85f36fe1afa2f27051d2b74a04e75cf7:6:18:0MPukV2BxRMEsQwkwEZ4shuxCnAUWWDRdAGv/1OjNlEI0sE+AR12H4DOdXre6tdj
cTweet:85f36fe1afa2f27051d2b74a04e75cf7:7:18:4QyC5ZpFSmOcoGB6WTNUJV/sBWX7Inn/Pup3iRVU9C4dnRkcx+6i7e6g/vfLo7kq
cTweet:85f36fe1afa2f27051d2b74a04e75cf7:8:18:wAqHm+t5Zq8z3Xlx3VqQtKp66zK5ceT2G3K6g4fBvyEbSiJlb+Nv/L7Ww/Uknf7j
cTweet:85f36fe1afa2f27051d2b74a04e75cf7:9:18:pA/JY/PgAaESyIDidcGMA9fIiF6SuOYD78t1j9LCKisnX8M0oLofNV7GHGoldMnm
cTweet:85f36fe1afa2f27051d2b74a04e75cf7:10:18:kWX0BqgbrOQrq85v1C3e97oPTcKM//ys0HDHXG/X8FFmGNafXfn

q78K0SJHChHg9
cTweet:85f36fe1afa2f27051d2b74a04e75cf7:11:18:JhzP6HlZVjVadFYYl/PFfNAFVsPPphNJQmK8xJes422bOdexwEVZ4BJ/LqnClUY6
cTweet:85f36fe1afa2f27051d2b74a04e75cf7:12:18:PPRvey5VGOKWgr7bdelnIg4I6stXBORhfdcVKiS13ApQm41Qh9iBVWtYnNcJrj7f
cTweet:85f36fe1afa2f27051d2b74a04e75cf7:13:18:o1zSagxja3wN9gMvKHL+2vjAaKHbH4anj9+9/x2frIwLFH1Qt2WCotU6UsZYqMr9
cTweet:85f36fe1afa2f27051d2b74a04e75cf7:14:18:gK3TCS4FtFAcGhKgTSP3+Gf5MAddXTLNNvF7Z70Cp9XjUAqI8+nwrxoHQVA7WC5G
cTweet:85f36fe1afa2f27051d2b74a04e75cf7:15:18:fN6tFBb1kQUzlwne03kVKFYQAkJC+NLYiGoztOwkE7xap91dEXnrAbr3mgLjcv5w
cTweet:85f36fe1afa2f27051d2b74a04e75cf7:16:18:iTsl0ULpjPQ1F6dkKOSn3kW0HvPwLGvFKbHBHGDxYrARhrCY/vGcm7Orl/VTNHF+
cTweet:85f36fe1afa2f27051d2b74a04e75cf7:17:18:iwHunoxJRgt8BUmabiFSQiRr
cTweet:85f36fe1afa2f27051d2b74a04e75cf7:18:18:=K7GqThe decryption is simple, too. The receiveencryptedwitter.pl script collects the direct messages containing the encrypted tweet. With the help of the line numbers and the msg-id the cyphertext will be reassembled, the number of total lines as stated in the direct messages is compared with the received number of lines for a message id to ensure all lines were received. As the msg id is the md5 has of the cyphertext, it would be a two- to threeliner to check the integrity of the reassembled cyphertext by calculating its m5 digest and compare it with the message id.

After the successful reassembly of the cyphertext, the script gathers the id of the secret key of the receiver, decrypts it and displays the encrypted message. #!/usr/bin/perl

```
use Net::Twitter;
use Crypt::GPG;
use Dumpvalue;

$user = "c0t0d0s0bob";
$password = "r78hpQbKA2vn9cu6Gy3R";

$identica = Net::Twitter->new(username=>$user, password=>$password, identica=>1);
$result = $identica->direct_messages();

@directmessages=@{$result};

foreach $i (@directmessages) {
    $speaker = $i->{sender_screen_name};
    $text    = $i->{text};
    $created_at = $i->{created_at};
    next unless $text =~ /cTweet/;
    ($msgid,$seqno,$total,$cuttext) = $text =~ /cTweet\:(.*)\:(.*)\:(.*)\:(.*)/;
    $ctweet{$msgid}->[$seqno] = $cuttext;
    $ctweet{$msgid}->[0] = $total;
    $ctweet_meta{$msgid}->{'speaker'}=$speaker;
    $ctweet_meta{$msgid}->{'created_at'}=$created_at;
    $ctweet_meta{$msgid}->{'lines'}++;
}

foreach $i (keys %ctweet) {
  $ctweetlength = $ctweet_meta{$i}->{'lines'};
  if ($ctweetlength == $ctweet{$i}[0]) {
   print "ctweet $i complete\n";
   $ctweet_complete{$i} = $ctweet{$i};
  } else {
   print "ctweet $i incomplete\n";
  }
}

my $gpg = new Crypt::GPG;
$gpg->gpgbin('/usr/bin/gpg');
```

```
$sec_keyline = `gpg --list-keys $user | grep pub`;
($secret_key) = $sec_keyline =~ /.*\/(.*) /;
$gpg->secretkey("$secret_key");
$gpg->passphrase("$password");

foreach $i (keys %ctweet_complete) {
 undef $ctweet_to_decrypt;
 $ctweet_to_decrypt = "-----BEGIN PGP MESSAGE-----\nVersion: GnuPG v1.4.6 (GNU/Linux)\n\n";
 for ($j=1;$j [0];$j++) {
   $ctweet_to_decrypt = $ctweet_to_decrypt . $ctweet{$i}->[$j] . "\n";
 }
 $ctweet_to_decrypt = $ctweet_to_decrypt . "-----END PGP MESSAGE-----\n";
 $encrypted = $ctweet_to_decrypt;
 ($ptweet,$signature) = $gpg->verify($encrypted);
 $speaker=$ctweet_meta{$i}->{'speaker'};
 $created_at=$ctweet_meta{$i}->{'created_at'};
 print "$speaker wrote at $created_at: $ptweet \n";
} When you start at first the ./sendencryptedtwitter.pl  and afterwards ./receiveencryptedtwitter.pl the second script script
delivers the unencrypted message. primary:/usr/src/cryptedtwitter# ./receiveencryptedtwitter.pl
ctweet 20390a191904ccf64d1eb571c3d7c887 complete
ctweet fdee6cdd401e69274e01376e8a4871d5 incomplete
c0t0d0s0alice wrote at Sun Feb 22 16:32:56 +0000 2009: Sed ut perspiciatis unde omnis
iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam,
eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo.
 Nemo enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia
consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt. Neque porro quisquam
 est, qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit, sed quia non numquam
 eius modi tempora incidunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim
ad minima veniam, quis nostrum exercitationem ullam corporis suscipit laboriosam, nisi ut aliquid
 ex ea commodi consequatur? Quis autem vel eum iure reprehenderit qui in ea voluptate velit esse
quam nihil molestiae consequatur, vel illum qui dolorem eum fugiat quo voluptas nulla pariatur?Message successfully
decrypted, proof of concept successful.
```

Posted by Joerg Moellenkamp in English, Privacy at 17:48

If you are going to deny that you are the intended recipient and use a public channel, then you best not use PGP/GPG in the event you are a suspect.

Your own machines are your worst enemy in this aspect and if used against you, you will likely be prosecuted.

It's best to keep it completely public, with a difficult "what you know" password or passphrase.
Anonymous on May 4 2009, 00:12

I believe it is possible to construct a straightforward method of communicating securely and relatively anonymously over the existing micro-blogging services, and have published an outline of how to do this.

http://www.caliban.org.uk/pmwiki/pmwiki.php?n=Main.SecureChannel
Anonymous on Jun 19 2009, 20:48

Just came across this post by complete chance. I'd wondered for a while if anyone had bothered to write an implementation of this. Nice work.
Anonymous on Apr 26 2010, 16:12