(A) Quick and Dirty Guide To a Fairly Secure Twitter Gateway by trem

Here I'll show you how to send tweets over the command line over SSH using perl script called TTYtter (http://www.floodgap.com/software/ttytter/) in Ubuntu.

This is specifically geared towards those living in countries where access to twitter is either monitored or forbidden. Ideally, this server would be hosted in a jurisdiction with friendlier laws regarding free speech and the media.

- Installing SSH
- Installing cURL/Lynx
- Installing TTYtter
- Tweeting

Installing SSH

From the command line, type the following

```
SUDO APT-GET UPDATE
```

```
trem@junky:~/.nothing/scripts/twitter$ sudo apt-get update
Get:1 http://security.ubuntu.com lucid-security Release.gpg [198B]
Ign http://security.ubuntu.com/ubuntu/ lucid-security/main Translation-en_CA
Ign http://security.ubuntu.com/ubuntu/ lucid-security/restricted Translation-en_CA
Hit http://archive.canonical.com lucid Release.gpg
Ign http://archive.canonical.com/ lucid/partner Translation-en_CA
Hit http://deb.torproject.org lucid Release.gpg
Hit http://ca.archive.ubuntu.com lucid Release.gpg
```

Next, we'll install the SSH daemon using the command

```
SUDO APT-GET INSTALL OPENSSH-SERVER
```

```
trem@junky:~/.nothing/scripts/twitter$ sudo apt-get install openssh-server
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  rssh molly-guard openssh-blacklist openssh-blacklist-extra
The following NEW packages will be installed:
  openssh-server
0 upgraded, 1 newly installed, 0 to remove and 46 not upgraded.
Need to get 0B/304kB of archives.
After this operation, 815kB of additional disk space will be used.
WARNING: The following packages cannot be authenticated!
  openssh-server
Install these packages without verification [y/N]? y
Preconfiguring packages ...
Selecting previously deselected package openssh-server.
(Reading database ... 192986 files and directories currently installed.)
Unpacking openssh-server (from .../openssh-server_1%3a5.3p1-3ubuntu5_amd64.deb) ...
Processing triggers for ureadahead ...
Processing triggers for ufw ...
```

Then, we'll start SSHD by typing the following:

```
SUDO /USR/SBIN/SSHD
```

Installing cURL/Lynx

We're going to need to install cURL or Lynx in order to use TTYtter. (for this tutorial, we'll install cURL) type the following command:

```
SUDO APT-GET INSTALL CURL
```

```
trem@junky:~$ sudo apt-get install curl
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  curl
0 upgraded, 1 newly installed, 0 to remove and 46 not upgraded.
Need to get 0B/210kB of archives.
After this operation, 336kB of additional disk space will be used.
WARNING: The following packages cannot be authenticated!
  curl
Install these packages without verification [y/N]? y
```

Installing TTYtter

Then, we'll need to download the TTYtter script using wget, then rename it. We'll do this by typing the following:

```
WGET HTTP://WWW.FLOODGAP.COM/SOFTWARE/
TTYTTER/DIST1/1.1.10.TXT
```

```
trem@junky:~/.nothing/scripts/twitter$ wget http://www.floodgap.com/software/ttytter/dist1/1.1.10.txt
--2011-02-19 02:35:11--  http://www.floodgap.com/software/ttytter/dist1/1.1.10.txt
Resolving www.floodgap.com... 66.159.214.137
Connecting to www.floodgap.com|66.159.214.137|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 150600 (147K) [text/plain]
Saving to: `1.1.10.txt'

100%[======================================================================================================================================>] 150,600     --.-K/s   in 0.06s

2011-02-19 02:35:17 (24.3 KB/s) - `1.1.10.txt' saved [150600/150600]

trem@junky:~/.nothing/scripts/twitter$
```

Let's rename the script something a little easier to remember. Type the following command.

```
MV 1.1.10.TXT TWITTER.PL
```

Then, we'll run the script by typing the following:

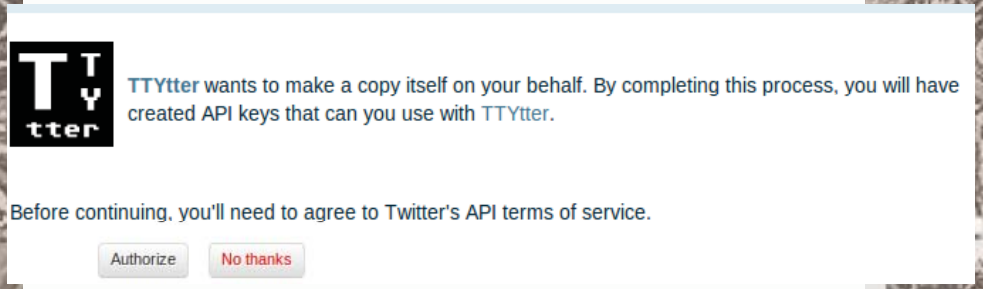
```
PERL TWITTER.PL
```

```
trying to find cURL ... /usr/bin/curl
-- no version check performed (use /vcheck, or -vcheck to check on startup)

++------------------------------------------------------------------++
||  WELCOME TO TTYtter: let's get you set up with an OAuth keyfile!  ||
++------------------------------------------------------------------++
Twitter now requires all applications authenticating to it use OAuth, a
more complex authentication system that uses tokens and keys instead of
screen names and passwords. To use TTYtter with this Twitter account,
you will need your own app key and access token. This requires a browser.

The app key/secret and user access token/secret go into a keyfile and
act as your credentials; instead of using -user, you use -keyf. THIS
KEYFILE NEVER EXPIRES. YOU ONLY NEED TO DO THIS ONCE FOR EACH ACCOUNT.

If you DON'T want to use OAuth with TTYtter, PRESS CTRL-C now. Restart
TTYtter with -authtype=basic to use a username and password. THIS IS
WHAT YOU WANT FOR STATUSNET, BUT WILL NOT WORK WITH TWITTER!
If you need help with this, talk to @ttytter or E-mail ckaiser@floodgap.com.

Otherwise, this wizard will create a keyfile /home/trem/.ttytterkey
for you. Press ENTER/RETURN to begin the process.
```

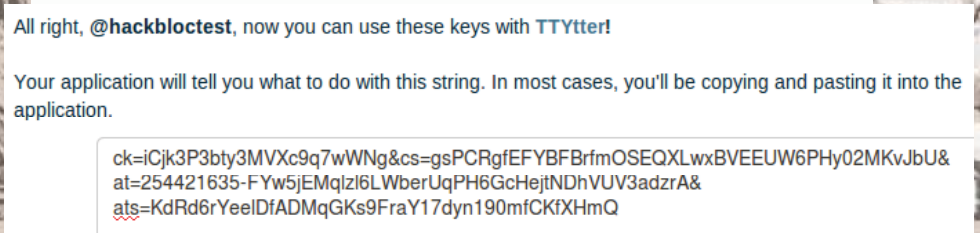
Press Enter or Return, and start Firefox and log into your twitter account. Copy and paste the URL into Firefox.

```
Start your browser.
1. Log in to https://twitter.com/ with your desired account.
2. Go to this URL (all one line). You must be logged into Twitter FIRST!

http://dev.twitter.com/apps/key_exchange?oauth_consumer_key=XtbRXaQpPdfssFwdUmeYw

3. Twitter will confirm. Click Authorize, and accept the terms of service.
4. Copy the entire string you get back.
-- Paste it into this terminal, then hit ENTER and CTRL-D to write it ---------
```

Click the 'Authorize' button



Now click the 'I Accept' button. On the next page, you will receive your key.



Copy and paste your key into TTYtter.

```
4. Copy the entire string you get back.
-- Paste it into this terminal, then hit ENTER and CTRL-D to write it --------
ck=iCjk3P3bty3MVXc9q7wWNg&cs=gsPCRgfEFYBFBrfmOSEQXLwxBVEEUW6PHy02MKvJbU&at=254
```

Hit enter then type CTRL-D

```
-- EOF ---------------------------------------------------------------
Written new key file /home/trem/.ttytterkey
Now, restart TTYtter to use this keyfile -- it will use this one by default.
(For multiple key files with multiple accounts, manually write them to other
 filenames, and tell TTYtter where the key is using -keyf=... . You can just
 use a text editor for that.)
```

This should be what you see if it was successfully installed.

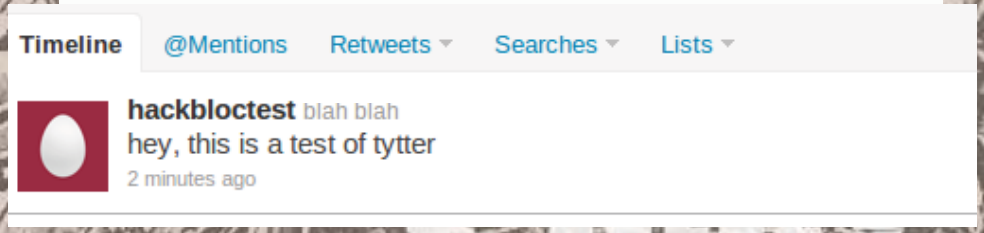
Tweeting

Finally let's run the script and send a test tweet. Type the following

```
PERL TWITTER.PL
```

```
##################################################        +oo=========oo+
        TTYtter 1.1.10 (c)2011 cameron kaiser             @             @
                 all rights reserved.                      +oo=   =====oo+
      http://www.floodgap.com/software/ttytter/          a==:  ooo
                                                         .++o++. ..o**O
  freeware under the floodgap free software license.     +++   :O:::::
       http://www.floodgap.com/software/ffsl/            +**O++ #   :ooa
                                                                  #+$$AB=.
         tweet me: http://twitter.com/ttytter                     #;;ooo;;
            tell me: ckaiser@floodgap.com                         #+a;+++;O
##################################################               ,$B.*o*** O$,
#                                                              a=o$*O*O*$o=a
# when ready, hit RETURN/ENTER for a prompt.                      @$$$$$@
# type /help for commands or /quit to quit.                       @o@o@o@
# starting background monitoring process.                         @=@ @=@
#
TTYtter> -- notification: API rate limit is currently 350 req/hr
-- no version check performed (use /vcheck, or -vcheck to check on startup)
-- you are logged in as hackbloctest

TTYtter> hey, this is a test of tytter
TTYtter>
```

From the TTYtter> prompt, here we can send our message to twitter.



Success!

How to use this server

This tutorial has showed you how to install SSHD and a command line twitter program. Utilizing these tools, users can send twitter messages over an encrypted connection. Shell hosting in the EU is fairly inexpensive.

example:

```
SSH <SERVERIPADDRESS>
```

then:

```
CD /PATH/TO/TTYTTERDIR

PERL TTYTTER.PL
```

Further reading:

https://hackbloc.org/content/how-plan-and-execute-act-electronic-civil-disobedience
https://hackbloc.org/sites/hackbloc.org/files/browservm.pdf
https://ssd.eff.org/
http://www.torproject.org/docs/tor-hidden-service.html.en
http://www.ubuntugeek.com/securing-ssh.html
http://www.howtoforge.com/truecrypt_data_encryption

ANONYMIZING LOGS WITH LOGROTATE BY CALL TO RUPTURE

Many service providers for social movements such as Indymedia make the choice to not keep logs that contain personally identifying information (PII). This is great, but this can create immense headaches for the many people behind these service providers. Imagine you run (or maybe you already do) a mail server for activists in the area. Somebody contacts you complaining that they are having problems receiving mail through their mail client. You have to parse through your logs but there isn't an obvious error message. You now have to find the logs of this particular person's connection but can't because you deleted all the personally-identifiable information before it even hit the disk.

For many system administrators, writing logs to an encrypted disk for a set period of time and then scrubbing them may be a more practical solution than never allowing PII into the logs. Unfortunately, this means that you have to figure out a way to scrub the logs on a schedule. Instead of looking for another piece of software to install on your server, you can use pre-existing utilities to do the job.

Most servers come with sed and logrotate installed. With these two utilities, we can scrub our logs on a schedule. Logrotate is a utility on Linux servers which manages log retention. According to the manpage, "logrotate is designed to ease administration of systems that generate large numbers of log files. It allows automatic rotation, compression, removal, and mailing of log files. Each log file may be handled daily, weekly, monthly, or when it grows too large. Normally, logrotate is run as a daily cron job." For example, /var/log/messages turns into /var/log/messages.1.gz then /var/log/messages.2.gz. At some point, it gets deleted. Logrotate is convenient because it's already managing your logfiles and it's a safe way to modify logs. Some programs are extremely picky about their logs and will malfunction or shutdown if they see that a log has been modified without their knowledge.

The first thing we'll need to do is find our logrotate config file. In the case of our system, it was in /etc/logrotate.conf. We can add our customizations there or we can create specific configs in /etc/logrotate.d. Regardless of which method you choose, you'll need to add customizations for each file we want to pull PII out of. For instance, here's what our config for our apache logs looks like. Comments added by the author are followed by // but don't try importing those into logrotate.

```
/var/log/apache2/*.log /var/log/apache2/ssl_error_log /var/log/apache2/
ssl_request_log { //Defines which logs we will be working on
        daily //rotate logs daily
        missingok //If the log file is missing, go on to the next one without is-
suing an error message.
        rotate 5 //Logs are only rotated five times after which they are deleted,
mailed out, etc
        compress //Compress logs when rotating them
        create 640 root adm //Create a new log with the following permissions
        sharedscripts //Only run postrotate once when rotation is done as op-
posed to during each rotation
        prerotate //Before rotating we run these commands to remove PII. This
section is critical. For each log this for statement finds, we run the sed state-
ment to pull out IPs and replace them with 0.0.0.0. Note that this will also
sanatize kernel version numbers so it's not perfect.
            for i in /var/log/apache2/*.log /var/log/apache2/ssl_error_log /var/
log/apache2/ssl_request_log
            do
              sed -i -e 's/[0-9]\{1,3\}\.[0-9]\{1,3\}\.[0-9]\{1,3\}\.[0-9]\
{1,3\}/0.0.0.0/g' $i
            done
        endscript //our scripting is done, this is an important line
        postrotate //do this after rotation
              if [ -f "`. /etc/apache2/envvars ; echo ${APACHE_PID_FILE:-/var/
run/apache2.pid}`" ]; then
                      /etc/init.d/apache2 reload > /dev/null
              fi
        endscript
}
```

Logrotate is a complicated beast with a lot of moving parts. It's easy to break and some problems are incredibly hard to diagnose. It took us around six hours to even figure out how to put that sed line in there. For this reason, only add/modify what is absolutely necessary and then do all your fun hacks later. The basic format used in the PII-remover script should be clear, so let's look at some other sed statements we have been using in our mail.log. These statements use regular expressions, which are hard to use at first but will significantly improve your sex life with sustained use.

Remove IPs sed -i -e 's/[0-9]\{1,3\}\.[0-9]\{1,3\}\.[0-9]\{1,3\}\.[0-9]\{1,3\}/0.0.0.0/g' $i
Remove IPs of our users sed -i -e 's/connect from [^]\+/connect from <anonymized_host>[0.0.0.0]/g' $i
Remove FROM addresses sed -i -e 's/from=<[^>]\+>/from=<anonymized_email_addr>/g' $i
Another FROM format sed -i -e 's/to=<[^>]\+>/from=<anonymized_email_addr>/g' $i
Remove usernames sed -i -e 's/user=[^,]\+,/user=anonymized_username,/g' $i

Thanks for reading and giving a damn about the privacy of your users.

Related Links:
Indymedia Wiki with Tips for System Admins https://docs.indymedia.org/Sysadmin/
Using Regular Expressions with sed http://www.tutorialspoint.com/unix/unix-regular-expressions.htm http://tinyurl.com/69hj988

Editor's note: There are certainly many other useful statements for removing PII in other logfiles. If you have some, send them in to staff{at}hackbloc.org for inclusion in the next issue.

PHONE HOME

Have you ever been in a situation where you got a box out in the "field" and you need to ssh into it, but chances are good it is going to be in a hostile environment? The people getting the computer start looking around the room for a kitten when you tell them to cat a file, the mention of the word firewall conjures up images of Bruce Willis in Die Hard 3 in the minds of your contacts on the receiving end, and you suspect the fucking pigs are gunna be subpoenaing all hosts and ips connected to from where your computer will be installed.

Have no fear, you now have the phone_home script to make that zombie call home with reverse ssh to another host running ssh on a tor hidden service.

```
# This is the startup script that will get the box callinghome on startup
# and restarting the connection if it is lost
#! /bin/sh

### BEGIN INIT INFO
# Provides:          phone_home
# Required-Start:    $remote_fs $syslog
# Required-Stop:     $remote_fs $syslog
# Default-Start:     2 3 4 5
# Default-Stop:      1
# Short-Description: OpenBSD Secure Shell server
### END INIT INFO

set -e

# /etc/init.d/ssh: start and stop the HB Phone Home daemon

test -x /root/bin/phone_home.pl || exit 0

. /lib/lsb/init-functions

# Are we running from init?
run_by_init() {
    ([ "$previous" ] && [ "$runlevel" ]) || [ "$runlevel" = S ]
}
```

```
export PATH="${PATH:+$PATH:}/root/bin"

case "$1" in
  start)
        log_daemon_msg "Starting HB Phone Home Daemon" "phone_
home.pl"
        if start-stop-daemon --start --quiet --oknodo --pidfile /var/run/
phone_home.pid --exec /root/bin/phone_home.pl ; then
            log_end_msg 0
        else
            log_end_msg 1
        fi
        ;;

  *)
        log_action_msg "Usage: /etc/init.d/phone_home {start}"
        exit 1
esac

exit 0

##### END of script

# And here is the actual phone home daemon
#!/usr/bin/perl
use strict;
use warnings;
use Getopt::Long;
use Sys::Syslog qw(:standard :macros);
openlog('phone_home daemon', 'nofatal', LOG_DAEMON);

# Instructions for a reverse ssh session are here:
# http://oogies.org/2009/07/08/reverse-ssh-tunneling/
# using tor has a tendency to forget about the host key:
# http://linuxcommando.blogspot.com/2008/10/how-to-disable-ssh-host-key-
checking.html
# Don't forget to set up public key authentication:
# http://sial.org/howto/openssh/publickey-auth/#s2
```

```
# parse command line options
my $username;
my $server;
GetOptions (
  'verbose+' => \$verbose,
  'user=s' => \$username,
  'server=s' => \$server,
);

if (!defined($username)) {
  print "Expecting a username.\n";
  usage();
  exit 0;
}

if (!defined($server)) {
  print "Expecting an onion server name to connect to.\n";
  usage();
  exit 0;
}

my $hidden_server = $username .'@'. $server;
my $ssh_command = '/usr/bin/ssh -R 9000:localhost:22 '.
  '-o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no '.

use POSIX qw(setsid sys_wait_h);

chdir '/'                 or die "Can't chdir to /: $!";
umask 0;
open STDIN, '/dev/null'   or die "Can't read /dev/null: $!";
open STDOUT, '>/dev/null' or die "Can't write to /dev/null: $!";
open STDERR, '>/dev/null' or die "Can't write to /dev/null: $!";
defined(my $pid = fork)   or die "Can't fork: $!";
exit if $pid;
setsid                    or die "Can't start a new session: $!";

# Check to make sure the phone home connection is running.
# Instructions for setting up ssh to a tor hidden service are here:
# http://ubuntu-unleashed.com/2008/03/howto-setup-anonymous-ssh-via-
```

```
tor-hidden-services.html
FORK:
syslog(LOG_INFO, "Phone Home. $ssh_command");
system($ssh_command);
my $exit = $?;
syslog(LOG_ERR, "ssh command exited with $exit. error: $!. $ssh_com-
mand");
sleep(5);
goto FORK;

sub usage {
print <<END;
$0 [options]
  --user <user>      The username on the remote server we should be con-
necting with.
  --server <server> The onion server name. ie. wmnw4fdnbdvzfcrg.onion
  --verbose          Increase the verbosity of this tool.
END
}
```

The Hidden Tracker Returns

The Hidden Tracker returned to normal operation this December. It seems like every day we hear about operators of BitTorrent trackers being harassed, arrested, and sued simply for providing the infrastructure people need to share files using the BitTorrent protocol. As a solution to this problem, a number of people have hosted trackers in countries with laws friendly to trackers only to have those countries (such as Sweden) bow to political pressure.

Decentralized tracking of torrents through DHT is slowly pushing trackers into obsolescence but the BitTorrent ecosphere does still rely to some extent on trackers. The Hidden Tracker is a project to run a publicly accessible tracker that doesn't have to worry about being shut down. The project utilizes the "hidden services" feature of Tor to anonymously host the website. This means that it's nearly impossible to find out where this site is hosted or shut it down. Note that this doesn't add any anonymity for downloaders/uploaders, just for the tracker itself.

If you are making a torrent, particularly one that is likely to be subject to censorship you might want to include this tracker. Just add these trackers to your torrent and voila! your torrent can't get shut down. One of the links uses the Tor2Web service which acts as a gateway to Tor which is accessible through the regular internet.

http://z6gw6skubmo2pj43.onion:8080/announce
https://kg6zclbtm7kwqide.tor2web.org/announce

Related links:
The Tor Project/Software: torproject.org
The Hidden Tracker Website: http://z6gw6skubmo2pj43.onion http://tinyurl.com/nvr5j4
via Tor2Web https://z6gw6skubmo2pj43.tor2web.org/ http://tinyurl.com/68fwros