Fighting XMPP messaging spam thanks to ejabberd API

process one
Leader in Messaging and Push Solutions

17th November 2015

Mickaël Rémond <mremond@process-one.net>

Introduction

There has been some discussions recently on XMPP operator mailing about the **rise of spam attacks on the XMPP federation**.

Posts on XMPP operators mailing list:

> **Simon Josefsson**
> ***Tue Nov 10 15:12:49 UTC 2015***
>
> Hi,
> I'm seeing similar spam, for me it started on October 27 and I've received spam on November 3rd, 4th, 7th and 9th as well. From different JIDs/servers.
>
> Is there any documentation on spam prevention on xmpp servers? Any system to submit spam examples that expose a DNS-based blacklists?
>
> I'm certain spamassassin would have scored the examples I received fairly high, but perhaps there are other ways to deal with spam in XMPP as well.
>
> /Simon

Reference: http://mail.jabber.org/pipermail/operators/2015-November/002745.html

Goal

I decided to add a talk to this ejabberd workshop to show ejabberd offer the needed tooling and API to fight attacks.

- Describe existing settings and tools
- Explore spam fighting tools improvements

Existing ejabberd anti-spam leverages

- Disable services that are obvious spam vectors
- Restrict abusive behaviour
- Block users and servers
- Analyse and detect issue to react

Disable or limit spammers "tools"

Handy tools for spammers are:

- User in-band registration - XEP-0077:
    - Allow self user creation from any client
    - Typically used to create massive number of accounts with bots
    - Accounts are used to send spam on local or remote servers through XMPP federation (s2s)
- Jabber User Directory - XEP-0055: Jabber Search:
    - Use to harvest XMPP account as spam target.

`mod_register` configuration best practices

- Make sure in-band registration is **disabled**. Check mod_register is not enabled / commented out in modules section of ejabberd:

```
modules:
  ...
  # mod_register: {}
...
```

- Or at least use **captchas** to protect registration from bots:

```
  hosts: ["example.org"]


  captcha_cmd:
"/lib/ejabberd/priv/bin/captcha.sh"
captcha_host: "example.org:5280"

listen:
  ...
  -
    port: 5280
    module: ejabberd_http
    captcha: true
...
```

Captcha challenge contains a link to use browser for client that do not support them directly.

- You can **restrict** registration to list of IPs:

```
acl:
  loopback:
    ip:
      - "127.0.0.0/8"
      - "::1"
access:
  mynetworks:
    loopback: allow
    all: deny
modules:
  ...
  mod_register:
    ip_access: mynetworks
...
```

- If you ever need to open in-band registration, at least use global level parameter to **limit number of registration per IP** per hour:

```
registration_timeout: 3600
```

Reference: [`mod_register` ejabberd documentation]

Web registration form

ejabberd even has **web registration** form with captcha, `mod_register_web`:

```
hosts: ["example.org"]
listen:
-
    port: 5281
    module: ejabberd_http
    register: true
    captcha: true
    certfile: "/etc/ejabberd/certificate.pem"
    tls: true
  ...
modules:
  ...
  mod_register_web: {}
  ...
captcha_cmd: "/lib/ejabberd/priv/bin/captcha.sh"
captcha_host: "example.org:5281"
```

You can then create account from https://example.org:5281

Reference: `mod_register_web` ejabberd documentation

Direct back-end integration for authentication

However, most development use **custom authentication** pointing to existing database.
Registration happens in back-end with typical existing spam prevention.

Policy change in ejabberd default configuration

Starting from ejabberd 15.12, `mod_register` will be disabled in default example configuration file.

User Directory

Unless you run an internal corporate server, you should probably disable Jabber Search feature, in `mod_vcard`
:

```
modules:
  ...
  mod_vcard:
    search: false
...
```

Reference: `mod_vcard` ejabberd Documentation

Policy change in ejabberd default configuration

Starting from ejabberd 15.12, `mod_vcard` search option default value will become false instead of true.

Restrict presence subscription request spam

`mod_presence_counter`

limits the sending or receiving of too many presence subscription requests:

```
modules:
  ...
  mod_pres_counter:
    count: 5
    interval: 60
...
```

Reference: [`mod_pres_counter` ejabberd documentation]

Server-to-server whitelist / blacklist

Even if you have enable a tight control on who can login on your own server, you can received spam messages coming from federated network.

ejabberd provides tools to allow admin to choose one of two modes of operation:

- **Whitelist:** open federation with only approved servers:

```
acl:
  trusted_servers:
    server:
      - "example.com"
      - "jabber.example.org"
access:
  s2s:
    trusted_servers: allow
    all: deny
s2s_access: s2s
...
```

- **Blacklist:** block certain servers from s2s federation if they have been identified as a source of spam:

```
acl:
  blocked_servers:
    server:
      - "example.com"
      - "jabber.example.org"
access:
  s2s:
    blocked_servers: deny
    all: allow
s2s_access: s2s
 ...
```

Multi-User Chat blocking spam and abuse

Multi-User Chat is often a privileged target for spammer as it amplifies the audience of each spam messages.

ejabberd has many tools to protect users against MUC spam:

- Blocking: MUC owners can ban members. This is standard XMPP.
- Preventing s2s users to join or just to create MUC:

```
acl:
  admin:
    user:
      - "admin": "example.org"
  local:
    user_regexp: ""
access:
  muc_admin:
    admin: allow
modules:
  ...
  mod_muc:
    access: all
    access_create: local
    access_persistent: local
    access_admin: muc_admin
...
```

- Many MUC traffic regulation options:
    - max_user_conferences: Maximum number a user can join at the same time to limit impact of bots.
    - min_message_interval
    - min_presence_interval
    - max_users_presence: overcrowded room management.

- Ability to limit identifiers: identifier can be used to spam or attempt to break client by using very long names:
    - max_room_id
    - max_room_name
    - max_room_desc
    - max_nickname: Missing option, will be added in upcoming ejabberd version

- Explicit anti spam options:
    - captcha_protected flag
    - spam_prevention flag for mod_muc_log

Other general tools available in ejabberd to prevent spam

- Reporting and metrics (through Teamleader or `mod_metrics` and Grapherl)
- Ability to sample usage during a few seconds to find abusers (Teamleader)
- Logging suspicious behaviour or abuse:
    - e.g. `mod_pres_counter` will log a warning when triggered.

What's next for spam prevention ?

Drafts XEP on Spam prevention:

- XEP-0159: Spim-Blocking Control
- XEP-0267: Server Buddies: Trust marker for peer servers.
- XEP-0268: Incident Handling: Allow Spam reporting, specifically between trusted servers.
- XEP-0275: Entity Reputation

ejabberd upcoming modules: Advanced Blocking

Thanks to it broad scope of API hooks, ejabberd is well equipped to let server admin build custom tools.

Block messages from contact not in roster

In XMPP message are not limited to users in your roster.

However, it makes sense to let server admin configure service to block messages from contact not in roster.

We need to extend `filter_packet`
API with information on roster, so that plugin can implement this filtering.

This rules could be enforced:

- only for contacts from other domains (message coming from federation)
- Both remote and local contact (more useful for public servers).

Roster subscription blocking

We can consider rejecting incoming contact addition on s2s if they have not been initiated by the local user.

This measure could be only temporary as if every server enables that option, it becomes impossible to add any user.

We can link the blocking to a reputation system, e.g. reject subscription request from user marked with low reputation from remote server.

Reputation system

It is possible to build a reputation system for users to give hints to other server about the user status.

For example a server could rely on user account creation date and activity to flag users as trusted.

This trust flag could be added to outgoing subscription requests.

Custom filter modules as response to spam pattern analysis

It is possible to use pattern seen on a current spam attack to block messages.

For example, on user on StackOverflow had his server targetted with standard message with subject.

As they are rarely used in standard chat approach, those messages can be blocked at least temporarily to block a certain spam attack.
Spammers are likely to change pattern, but it can help mitigate issue.

In that case, spam is annoying because some clients open special pop up message window for each message with a topic.

Here is for example a filter module targeting that case:

```
%% Declare module in ejabberd.yml
%% modules:
%%   mod_subject_block: {}
-module(mod_subject_block).

-behaviour(gen_mod).

-include("../deps/p1_xml/include/xml.hrl").
-include("logger.hrl").

-export([start/2, stop/1,
         on_filter_packet/1]).

start(_Host, _Opts) ->
    ejabberd_hooks:add(filter_packet, global, ?MODULE, on_filter_packet, 50),
    ok.

stop(_Host) ->
    ejabberd_hooks:delete(filter_packet, global, ?MODULE, on_filter_packet, 50),
    ok.

on_filter_packet({_From, _To, XML = #xmlel{name = <<"message">>, attrs = Attrs}} = Arg) -
>
    ?INFO_MSG("Filtering message ~p", [XML]),
    case proplists:get_value(<<"type">>, Attrs, <<"normal">>) of
        <<"normal">> ->
            drop;
        _ ->
            check_subject(Arg)
    end;
on_filter_packet(Packet) ->
    Packet.

check_subject({From, To, XML = #xmlel{name = <<"message">>, children = Children}}) ->
    ShouldBlock = lists:foldl(fun(#xmlel{name = <<"subject">>}, _) -> true;
                                 (_Child, State) -> State
                              end, false, Children),
    case ShouldBlock of
        true ->
            drop;
        _ ->
            {From, To, XML}
    end.
%%...
```

Conclusion

During the past years, XMPP has been relatively free from spam.

Most of the issue were Denial of Service under various forms.

However, the recent rise of spam is make spam fighting an important topic on ejabberd development roadmap.

You can expect more modules to fight spam in the short term.