Yandex.Root #2

Igor Gnatenko (Linuxorgru)

April 15, 2015

Contents

I Приготовления и запуск 1

II Задачи 3

1 Echo 4

2 Git 5

3 Mail 9

4 TwMail 13

5 Exec 14

6 DB repl 15

7 Balancer 16

8 Jabber 17

9 Jabber Archive 18

III Итоги 19

Abstract

Part I

Приготовления и запуск

Сразу редактируем grub cmdline, добавляя init=/bin/bash. Запускаем. Не работает ввод (или через раз, у кого как). Втыкаем в параметры загрузки, убираем странный vconsole.keymap=us, загружаемся, УРА. Перемонтируем в rw, меняем пароль руту. Запускаем. При попытке логина SELinux блокирует доступ к /etc/shadow :(Дальше мнения расходятся, кто-то просто выключил селинукс (selinux=0 в grub), кто-то запустил переразметку (touch /.autorelabel). По поводу firewall мнения опять разошлись, кто-то отключил firewalld, кто-то настроил ;).
Дальше как обычно, yum install wireshark tcpdump bind-utils screen.
Полчаса убили на развёртывание общей VM и выкачивание ssh ключей.

Part II

Задачи

Chapter 1

Echo

Set up an echo server on port 2000.

У яндекса в течении 2х часов то работало, то не работало. Никаких пакетов не слышали. Tailgunner выдал идею, что возможно приходят sctp пакеты. Запускаем tshark sctp. Тишина. Потом, когда VPN стабилизировался, побежали sctp пакетики. Стали искать SCTP Echo Server. Delirium нагуглил. Запустили. Не засчитывает потому, что в это время пытаемся настроить DNS и на одной из задач он адово виснет. Пришлось поднять на локалхосте сервер и засчитано.

Chapter 2

Git

Set up git-server with access via http://yandex:root@your_ip/log.git.
If someone create a file with .appendonly extention, the only opperation than allowed for such is file is append.

```
# yum install git nginx fcgi-devel spawn-fcgi -y
# git clone git://github.com/gnosek/fcgiwrap.git
# cd fcgiwrap
# autoreconf -i
# ./configure
# make
# cp fcgiwrap /usr/local/bin/
# vi /etc/nginx/nginx.conf
# mkdir /srv/git
# cd /srv/git/
# git init --bare --shared log.git
# chown -R nginx:nginx log.git
# vi log.git/hooks/pre-receive
# chmod +x log.git/hooks/pre-receive
# vi ~/fcgi.sh
# ~/fcgi.sh start
# systemctl start nginx
```

```
#!/bin/bash

validate_ref2() {
    oldrev=$(git rev-parse $1)
    newrev=$(git rev-parse $2)
    refname="$3"

    LIST=`git diff --name-only $oldrev $newrev | grep -P '^.*\.ao'`

    for file in $LIST; do
        if  git diff $oldrev $newrev -- $file | grep -v -P '^(---|\+\+\+)' | grep '^-' > /dev/null ; then
            echo "Only append is allowed for .append!"
            exit 1
        fi
        if  git diff $oldrev $newrev -- $file | grep -v -P '^(---|\+\+\+)' | pcregrep -M '^\+(.*\n)*\s' > /dev/null ; then
            echo "Only append is allowed for .append!"
            exit 1
        fi
    done
}

fail=""

if [ -n "$1" -a -n "$2" -a -n "$3" ]; then
    PAGER= validate_ref2 $2 $3 $1
else
    while read oldrev newrev refname
    do
        validate_ref2 $oldrev $newrev $refname
    done
fi

if [ -n "$fail" ]; then
    exit $fail
fi
```

```
root /srv/git/;
location ~ ^/([^/]+\.git)(/.*|$) {
        include fastcgi_params;

        fastcgi_param PATH_INFO $uri;
        fastcgi_param REMOTE_USER $remote_user;
        fastcgi_param SCRIPT_FILENAME /usr/libexec/git-core/git-http-backend;
        fastcgi_param GIT_PROJECT_ROOT $document_root;
        fastcgi_param GIT_HTTP_EXPORT_ALL "";

        fastcgi_pass unix:/var/run/git-fcgi.sock
}
```

Listing 2: nginx.conf

```
#!/bin/sh
#
# git-fcgi   The Git HTTP/FastCGI server
#
# chkconfig: - 80 20
# processname: git-fcgi
# description: Git HTTP/FastCGI server
# pidfile: /var/run/git-fcgi.pid

### BEGIN INIT INFO
# Provides: git-fcgi
# Required-Start: $local_fs $remote_fs $network
# Required-Stop: $local_fs $remote_fs $network
# Default-Start: 2 3 4 5
# Default-Stop: 0 1 6
# Short-Description: Start and stop Git HTTP/FastCGI server
### END INIT INFO

# Source function library.
. /etc/init.d/functions

# Config & Vars
prog=git-fcgi
childs=1
pidfile=/var/run/git-fcgi.pid
lockfile=/var/lock/subsys/git-fcgi
sockfile=/var/run/git-fcgi.sock
sockmode=0700;
sockuser=nginx
sockgroup=nginx
proguser=nginx
proggroup=nginx
gitexec=/usr/libexec/git-core/git-http-backend
fcgiexec=/usr/local/sbin/fcgiwrap
spawnexec=/usr/bin/spawn-fcgi
progexec="${spawnexec} -u ${proguser} -g ${proggroup} -U ${sockuser} -G ${sockgroup} -P ${pidfile} -s ${sockfile} -M ${sockmode} -- ${fcgiexec} -f -c ${childs} -p ${gitexec}"
RETVAL=0

# Functions
start() {
    echo -n $"Starting ${prog}: "
    [ -n "${sockfile}" -a -S "${sockfile}" ] && rm -f ${sockfile}
    daemon "${progexec} > /dev/null"
    RETVAL=$?
    echo
    [ $RETVAL = 0 ] && touch ${lockfile}
    return $RETVAL
}

stop() {
    echo -n $"Stopping ${prog}: "
    [ -n "${sockfile}" -a -S "${sockfile}" ] && rm -f ${sockfile}
    killproc -p ${pidfile} ${prog}
    RETVAL=$?
    echo
    [ $RETVAL = 0 ] && rm -f ${lockfile} ${pidfile}
    return $RETVAL
}

restart() {
    stop
    start
}

reload() {
    restart
}

force_reload() {
    restart
}

rh_status() {
    status -p ${pidfile} ${prog}
}

# Main
case "$1" in
    start)
        rh_status > /dev/null 2>&1 && exit 0
        start
        ;;
    stop)
        stop
        ;;
    status)
        rh_status
        RETVAL=$?
        ;;
    restart)
        restart
        ;;
    reload)
        reload
        ;;
    force-reload)
        force_reload
        ;;
    condrestart|try-restart)
        if rh_status > /dev/null 2>&1; then
            restart
        fi
        ;;
    *)
        echo $"Usage: $prog {start|stop|restart|reload|force_reload|condrestart|try-restart|status|help}"
        RETVAL=2
esac

exit $RETVAL
```

Listing 3: fcgi.sh

Делал это всё realloc. Общую виртуалку как-то испоганили, пришлось на локалхосте поднимать и заново всё делать.

Output: Error: [Errno 13] Permission denied: '/root'

При этом всё работало, но у Яндекса снова начались проблемы с VPN и game. Потеряли полтора часа из-за этого.

Chapter 3

Mail

Set up mail server. SMTP and IMAP (with SSL)
We will create new mail users using login 'amu' and cmd sudo -n amu $username $password

router:
ВАЖНО: задача из серии "хочу чтобы всё было хорошо и ещё мир во всём мире. детали сам придумаешь". Приведённое решение кривое и уязвимое, т.к. цель была - угадать, какие требования робот-проверяльщик предъявляет к сервису. Не надо бездумно применять его на реальных серверах.

1. в visudo разрешил безпарольное выполнение команд для группы wheel и отключел требование tty

```
#Defaults    requiretty
%wheel  ALL=(ALL)       NOPASSWD: ALL
```

   З.Ы. а вообще-то это не нужно было :)

2. установил postfix (smtp, smtps) и dovecot (pop3, pop3s, imap, imaps). в конфигах dovecot (цитирую только то, что менял и что существенно для задачи) :

```
# imap, imaps, pop3, pop3s в конфиге dovecot включены по дефолту
# вместо системных пользователей нам нужны виртуальные, существующие только для почтовой системы.
# Поэтому вместо PAM используем файл с паролями и фиксированные UID:GID
mail_location = maildir:/opt/mail/mail/%n/Maildir
passdb {
  driver = passwd-file
  args = scheme=CRYPT username_format=%u /opt/mail/db/users.txt
}
userdb {
  driver = static
  args = uid=1003 gid=1003 home=/opt/mail/mail/%u allow_all_users=yes
}
# принимаем запросы на аутентификацию через unix сокет.
# Postfix будет аутентифицировать пользователей через dovecot sasl, используя этот сокет
service auth {
  /etc/dovecot/conf.d/10-master.conf:  unix_listener auth-userdb {
  }
  unix_listener /var/spool/postfix/private/auth {
    mode = 0666
  }
}
# самоподписанный сертификат. Как и прошлый раз,
# Subject: C=RU, ST=Moscow, L=Moscow, O=Yandex, OU=Root, CN=10.0.0.91/emailAddress=mail@yandex.com
# хотя на этот раз сертификат никто с пристрастием не проверял
ssl_cert = </etc/ssl/certs/10.0.0.91.crt
ssl_key = </etc/ssl/keys/10.0.0.91.key

# необходимо для работы postfix LDA. адрес произвольный, главное чтобы почтовая система хоть куда-нибудь успешно доставляла эту почту
postmaster_address = root@localhost

# т.к. робот хочет в т.к. pop3, imap без ssl. Небезопасно
disable_plaintext_auth = no
login_trusted_networks = 10.0.0.0/16
```

   Теперь postfix. цитаты из main.cf

```
# ещё раз, подбирал методом тыка по логам
myhostname = davies
mydomain = root

# доставлять будем через virtual транспорт dovecot
virtual_mailbox_domains = root, davies.root, root.root
```

```
virtual_transport = dovecot

# это для следующей части задания
virtual_alias_maps = hash:/etc/postfix/virtual

# аутентификация через dovecot sasl, через unix сокет
smtpd_sasl_type = dovecot
smtpd_sasl_path = private/auth
smtpd_sasl_auth_enable = yes

# включаем ssl и tls
smtpd_use_tls = yes
smtpd_tls_key_file = /etc/ssl/keys/10.0.0.91.key
smtpd_tls_cert_file = /etc/ssl/certs/10.0.0.91.crt
smtpd_tls_CApath = /etc/ssl/certs
smtpd_tls_loglevel = 2
smtpd_tls_received_header = yes

# небезопасно
disable_plaintext_auth = no

# небезопасно! за применение на реальном сервере отрывать руки!
# после аутентификации пусть хоть весь яндекс заспамят. Но анонимусы идут лесом. Регистратов-то можно поймать и повесить
smtpd_recipient_restrictions =
        permit_sasl_authenticated
        reject

# хм, интересно, что это и зачем я это включил
dovecot_destination_recipient_limit = 1
```

Создаём пользователя, который фактически будет владельцем почтовых ящиков. Именно эти UID:GID используются dovecot'ом (userdb)

```
# id fakemailuser
uid=1003(fakemailuser) gid=1003(fakemailuser) группы=1003(fakemailuser)
```

Создаём тестового пользователя для ручной проверки через telnet / openssl s_client

```
# doveadm pw -p "password" -s crypt
{CRYPT}2e6EW/cpsCaDU
# echo 'user:{CRYPT}2e6EW/cpsCaDU' > /opt/mail/db/users.txt
```

и кодируем их для проверки AUTH PLAIN (postfix)

```
# perl -e 'use MIME::Base64; print encode_base64("\000user\000password");'
AHVzZXIAcGFzc3dvcmQ=
```

Проверяем

imap
```
telnet localhost 143
[...]
. login user password
```

imaps
```
openssl s_client -connect localhost:993
[...]
. login user password
```

pop3
```
telnet localhost 110
[...]
user user
[...]
pass password
```

pop3s
```
openssl s_client -connect localhost:995
[...]
user user
[...]
pass password
```

smtp
```
telnet localhost 25
[...]
ehlo localhost
auth plain AHVzZXIAcGFzc3dvcmQ=
```

smtps
```
openssl s_client -connect localhost:465
openssl s_client -connect localhost:25 -starttls smtp
[...]
ehlo localhost
auth plain AHVzZXIAcGFzc3dvcmQ=
```

3. Теперь вспоминаем, что робот будет ломиться по ssh для создания пользователей, причём вместо команды будет сразу кидать "login" "password" поэтому пишем скрипт, который назначим ему вместо shell

```
# tail -n2 /etc/shells
/usr/local/bin/makemailuser.pl
/usr/bin/perl
```

```
#!/usr/bin/perl
# всегда
use strict;
use Data::Dumper;

# PATH
$ENV{'PATH'}='/bin:/usr/bin:/sbin:/usr/sbin';

# собираем то, что робот нам прислал
my $credentials = $ARGV[1];
# разбиваем на отдельные слова
my @tmp = split ( ' ', $credentials );
# нас интересует 4 и 5 слово ( в perl нумерация с 0 )
my ( $user, $cleartext_password ) = ( $tmp[3], $tmp[4]);

# убираем кавычки
$user =~ s/^"//;
$cleartext_password =~ s/^"//;
$user =~ s/"$//;
$cleartext_password =~ s/"$//;

# здесь непосредственно добавляем пользователя в файл для dovecot
sub add_user_to_db($$){
	my $user = shift;
	my $cleartext_password = shift;

	# шифруем в CRYPT
	# TODO: rewrite with native perl module
	my $passwd = `doveadm pw -p "$cleartext_password" -s crypt `;
	chomp $passwd;

	# открываем, дописываем
	my $userdb = '/opt/mail/db/users.txt';
	open FILE, ">>", $userdb;
	#printf FILE "%s\n", join(';', @ARGV);
	printf FILE "%s:%s\n", $user, $passwd;
	close FILE;
}

# это для следующей задачи
# добавляем в /etc/postfix/virtual маппинги вида
# @user  user
# @user.root user

# param:
# - user
sub add_mapping($) {
	my $user = shift;
	my $map = '/etc/postfix/virtual';

	# добавляем
	open MAP, ">>", $map;
	printf MAP "\@%s %s\n", $user, $user;
	printf MAP "\@%s.root %s\n", $user, $user;
	close MAP;

	# и выполняем postmap
	`sudo postmap $map`;
}

# просто пишем в лог /tmp/amu_log.txt
sub log_this_shit($){
	my $str = shift;

	my $filename = '/tmp/amu_log.txt';
	open FILE, ">>", $filename;
	#printf FILE "%s\n", Dumper(\@ARGV);
	printf FILE "%s\n", $str;
	close FILE;
}

# поехали
if ( ( $tmp[0] eq 'sudo' ) and ( $tmp[1] eq '-n' ) and ( $tmp[2] eq 'amu' )) {
	# поступила команда "добавить пользователя"

	add_user_to_db($user, $cleartext_password);
	add_mapping($user);
} else {
	# ты тут самый умный, да? сам догадался или сказал кто?

	# робот тихой сапой шлёт команды на добавление правила файрвола
	my $cmd = join (' ', $credentials );
	log_this_shit($cmd);

	# и на это нужно возвращать ошибку. Иначе робот будет не зачтёт задание
	exit 1
}
exit 0; # на всякий случай
```

Listing 4: makemailuser.pl

содержимое /opt/mail/db/users.txt после нескольких запусков game:

```
70802fbae2:{CRYPT}dPmSaTyol6BhU
e82216943d:{CRYPT}Dmqh9EYmsFNW.
[...]
cccedc3d60:{CRYPT}jCPOWgEuFPymc
```

содержимое /tmp/amu_log.txt после нескольких проверок:

```
sudo -n iptables -I INPUT -p tcp -m multiport --dports 25,587 -j REJECT >/dev/null 2>/dev/null
sudo -n iptables -I INPUT -p tcp -m multiport --dports 25,587 -j REJECT >/dev/null 2>/dev/null
[...]
sudo -n iptables -I INPUT -p tcp -m multiport --dports 25,587 -j REJECT >/dev/null 2>/dev/null
```

Chapter 4

TwMail

Same as "Mail" task, but SMTP and POP3.
Additionally, the users should be available to receive mail using "@login" address.

router:
smtp и pop3, pop3s уже настроены в предыдущем пункте
сложность именно в том, чтобы принимать почту на некорректные адреса (кто-нибудь знает - зачем? первый раз вижу такой изврат)
для этого в

- postfix добавлено virtual_alias_maps (3)
- в скрипт /usr/local/bin/makemailuser.pl добавлена функция add_mapping (3)

проверяющий робот тупит и принимает задачу только со второй проверки

Chapter 5

Exec

We have got two very old programs. Run it on the same machine.

Подсказываю delirium'у, где лежат 2 бинарника, дальше он всё делает очень быстро.

```
# cd /opt/archive/
# file s1 s2
s1: ELF 32-bit MSB executable, SPARC, ... , not stripped
s2: ELF 32-bit MSB executable, PowerPC or cisco 4500, ..., not stripped
# yum install qemu-user
# screen qemu-sparc s1
# screen qemu-ppc s2
```

Chapter 6

DB repl

We have got a DB on port 5984. Unfortunately, one day the replication has been broken, and one client write some data to slave DB.
This is very critical DB, so fix the replication, and save only the last version of each document.

Быстро нагугливаем, что обычно живёт на 5984 порту. Ага, CouchDB. Ставим, чиним права на файлы, стартуем. Долго втыкаем что хотят от нас. Пока я занимался жаббером, кочдб тыкали все, кто только дотянулся. Роутер доделывает почту, я присоединяюсь к решению базы. Анализирую документы, потому что octocat смерджил мастер/слейв. Вижу, что поле "t" в каждом документе - это UNIX timestamp. Подтянулся lumi. Приходит идея, что нужно брать не последний номер документа (id), а смотреть по этому таймштампу. Восстанавливаем оригинальные базы.
Пишу скрипт на питоне, делаем репликацию.
Было сделано последним.

```python
#!/usr/bin/env python
import datetime
import requests
import json

host = "http://127.0.0.1:5984"
headers = {'Content-type': 'application/json'}
js = requests.get("{}/words_slave/_all_docs?conflicts=true".format(host)).json()
requests.put("{}/w".format(host))
for l in js["rows"]:
    master = requests.get("{}/words/{}".format(host, l["id"])).json()
    master_ts = datetime.datetime.fromtimestamp(master["t"])
    slave = requests.get("{}/words_slave/{}".format(host, l["id"])).json()
    slave_ts = datetime.datetime.fromtimestamp(slave["t"])
    if master_ts > slave_ts:
        requests.put("{}/w/{}".format(host, l["id"]), data=json.dumps(master), headers=headers)
    else:
        requests.put("{}/w/{}".format(host, l["id"]), data=json.dumps(slave), headers=headers)
```

Listing 5: couch-repair.py

```
# mkdir /var/log/couchdb
# chown couchdb:couchdb /var/log/couchdb
# systemctl start couchdb
# ./couch-repair.py
# systemctl stop couchdb
# cd /var/lib/couchdb/
# mv w.couch words.couch
# rm words_slave.couch -f
# systemctl start couchdb
# export HOST="http://127.0.0.1:5984"
# curl -X PUT $HOST/words_slave
# curl -H "Content-Type: application/json" \
      -X POST $HOST/_replicate \
      -d '{"source":"words","target":"http://127.0.0.1:5984/words_slave"}'
```

Chapter 7

Balancer

You have got the daemons on the ports 9001-9005.
Set up a balancer on port 9000 which will be evenly balance connection to the daemons.

tazhate:
netstat - nlp, ага оно tcp, далее обычный haproxy с тупым и банальным конфигом.

```
# yum install haproxy -y
# vi /etc/haproxy/haproxy.cfg
# systemctl start haproxy
```

```
listen proxy :9000
      mode tcp
      option tcplog
      balance leastconn
      server demon1 127.0.0.1:9001 check
      server demon2 127.0.0.1:9002 check
      server demon3 127.0.0.1:9003 check
      server demon4 127.0.0.1:9004 check
      server demon5 127.0.0.1:9005 check
```

Listing 6: haproxy.cfg

Зная ключевое слово haproxy задача решается натурально за 5 минут. если бы не глюки яндекса и начальная суматоха...

Chapter 8

Jabber

Some time ago there was a jabber server for root.yandex.net on the machine. Make it work again.
You can find some useful data in /var/lib/ejabberd

Сразу начинаем настраивать bind. В фоне кто-то ставит ejabberd. Час пытались настроить зону, валидатор не проходил. Взял зону со своего сервера, конфиг взял дефолтный из пакета. OpenVPN отваливается, айпишники меняются, приходится каждый раз править зону :(. Настроили SRV для xmpp-server, xmpp-client - game не проходит. Запускаю tcpdump -v -XX proto udp port 53, пытаюсь понять. Один из пакетов выглядит так:

```
13:31:54.403522 IP (tos 0x0, ttl 64, id 58620, offset 0, flags [none], proto UDP (17), length 74)
    10.0.16.1.38146 > 10.0.17.244.domain: [udp sum ok] 36822+ SRV? _jabber._tcp.root.yandex.net. (46)
        0x0000:  3e80 ee1c 4cf2 5682 683d aab1 0800 4500  >...L.V.h=....E.
        0x0010:  004a e4fc 0000 4011 5fb2 0a00 1001 0a00  .J....@._.......
        0x0020:  11f4 9502 0035 0036 7866 8fd6 0100 0001  .....5.6xf......
        0x0030:  0000 0000 0000 075f 6a61 6262 6572 045f  ......._jabber._
        0x0040:  7463 7004 726f 6f74 0679 616e 6465 7803  tcp.root.yandex.
        0x0050:  6e65 7400 0021 0001                      net..!..
```

Выплёвываю мысль, что SRV запись не та, меняю запись на нужную - всё понеслось. В фоне imul донастроил ejabberd, но всё равно не работает.

```
$TTL 86400
@ IN SOA ns.yandex.net. root.yandex.net. (
        2015040101;
        86400;
        7200;
        2419200;
        10800;
)

@                       86400   IN      NS      ns
root.yandex.net.        86400   IN      A       10.0.34.45
ns                      86400   IN      A       10.0.34.45
_xmpp-client._tcp.yandex.net.       86400 IN SRV 5 0 5222 root.yandex.net.
_xmpp-server._tcp.yandex.net.       86400 IN SRV 5 0 5269 root.yandex.net.
_xmpp-client._tcp.root.yandex.net.      86400 IN SRV 5 0 5222 root.yandex.net.
_xmpp-server._tcp.root.yandex.net.      86400 IN SRV 5 0 5269 root.yandex.net.
_jabber._tcp.root.yandex.net.       86400 IN SRV 5 0 5269 root.yandex.net.
```

Смотрю конфиг ejabberd - там что-то про ssl. Первая мысль - отключить. Комментирую в ямле module: ejabberd_c2s, starttls: true. Ребутаем это поделие на эрланге - пройдено. ejabberd был взят с офсайта и нагло установлен в /opt/ejabberd %)

Chapter 9

Jabber Archive

Make all the jabber messages dumped to http://ip/jabber_archive.txt (any text format)

После поднятия жаббера перемещаемся сюда. anonymous_sama нашёл модуль mod_log_chat, который должен был сделать всё что надо.
Lumi пишет конфигт долго и безуспешно пытается откомпилировать модуль.

```
mod_log_chat:
  path: "/srv/git/jabber_archive.txt"
  format: text
```

Казалось бы, 5 минут и решение готово, но оказалось не так всё просто. Нагуглил svn репозиторий с этим модулем и, не прочитав ворнинг об устаревании репа, клонирую, компилирую. Подкладываем в ebin/ - падает сервер при запуске =(. И так и сяк, не хочет работать, роем гугл, пробуем разное. Ошибка странное, ничего не понятно (никто с эрлангом дела никогда не имел). Замечаю ворнинг на сайте проекта и что сейчас нужно использовать git, а не svn репозиторий. Клонирую, компиляю, кладу в ebin, сервер запускается. Но при попытке теста через game в error.log появляется

```
2015-04-14 15:17:28.875 [error] <0.460.0>@ejabberd_hooks:run1:335 {{case_clause,{error,enotdir}},[
{mod_log_chat,write_packet,4,[{file,"mod_log_chat.erl"},{line,146}]},
{ejabberd_hooks,safe_apply,3,[{file,"src/ejabberd_hooks.erl"},{line,385}]},
{ejabberd_hooks,run1,3,[{file,"src/ejabberd_hooks.erl"},{line,332}]},
{ejabberd_c2s,session_established2,2,[{file,"src/ejabberd_c2s.erl"},{line,1299}]},
{p1_fsm,handle_msg,10,[{file,"src/p1_fsm.erl"},{line,582}]},
{proc_lib,init_p_do_apply,3,[{file,"proc_lib.erl"},{line,239}]}
]}
```

В ejabberd.log:

```
2015-04-14 15:33:21.456 [error] <0.634.0>@mod_log_chat:open_logfile:173 Cannot write into file
/srv/git/jabber_archive.txt/2015-04-14 alice@root.yandex.net - bob@root.yandex.net.log: enoent
```

enotdir наводит меня на мысль, что он хочет директорию, а ему суют файл. Меняю в конфиге path на /srv/git/ - УРА! Но не тут то было. Яндекс хочет получить сообщения по http://ip/jabber_archive.txt, а у нас создаётся каждый раз при запуске game 2015-04-14 alice@root.yandex.net - bob@root.yandex.net.log. Решаю симлинком в /srv/git/. nginx уже сервит эту директорию.

Part III

Итоги

9 место. Огромное спасибо tazhate, madrouter, realloc, lumi, octocat, delirium, anonymous_sama, всем кого забыл, ну и мне %)

Echo	Git	Mail	TwMail	Exec	DB repl	Balancer	Jabber	Jabber Archive
02:09:29	04:13:41	06:02:54	07:25:07	00:45:43	08:50:41	00:42:44	04:53:19	06:38:24