

# Yandex.Root Final

Igor Gnatenko (Linuxorg.ru)

April 23, 2015

# Contents

I	Приготовления и запуск	1
II	Задачи	3
1	Backup	4
2	HTTPS MITM	5
3	CI	6
4	netflow	7
5	Repo	8
6	Infected binary	9
7	DNS MITM	10
8	SVN	12
9	NFS	13
10	Nginx Lua	15
III	Итоги	17

## Abstract

## Part I

# Приготовление и запуск

2 образа. Debian, OpenIndiana. Всё как обычно, меняем пароль руту, грузимся.

Part II

Задачи

# Chapter 1

## Backup

Our old admin supported small site with some very useful texts. Unfortunately, he quit some time ago and we do not know how to restore the site. Please make it run again with the most recent data available.

true\_admin:

Сразу смотрим `/.bash_history` пока его не затёрли (`HISTSIZE=10!`).

Оттуда понятно что бэкапы лежат в `/var/backups` и так же виден ключ и то что шифровалось через `openssl`. Попутно выясняется что ключ подходит только к первому архиву. А тааам... ключ ко второму архиву в котором... ну вы поняли.

```
HISTSIZE=10000 иии...
cd /var/backups
KFILE=backup.key
TMPFILE=/tmp/temp.arch
# самый первый ключ из .bash_history
echo 27CBB6C8170EF54C3235AF3F8ED2A106 > $KFILE

# небольшой уикл
for f in $(ls *encrypted | sort -r); do
    newname=$(basename $f .encrypted);
    openssl aes-128-cbc -d -kfile $KFILE -in $f -out $newname;
    tar --strip-components=1 -xf $newname root/backup.key
done

for f in $(ls *tgz | sort); do tar -xf $f; done
# копируем usr/share/nginx/www в корень.
mv usr/ /
```

## Chapter 2

# HTTPS MITM

We continue with our Internet filtration topic. Now you need to set up HTTP proxy on port 3129 which will intercept all HTTPS traffic coming through and do it properly: we expect the proxied sites to provide valid certificates for their associated host-names. You may sign it with our own MITM certificate authority (the CA key and certificate are available in /root), we will use a client that trusts this CA.

To additionally prove that you are ready to intercept traffic, please substitute the value for Server: header of all requests with the string "root.yandex.com".

Ставим mitmproxy, запускаем и останавливаем для того чтобы сгенерировались сертификаты  
потом `cd /root ; cat mitm.key mitm.crt > .mitmproxy/mitmproxy-ca.pem`

Затем:

```
1 def response(context, flow):
2     flow.response.headers["Server"] = ["root.yandex.com"]
```

Listing 1: add\_header.py

```
mitmproxy -p 3129 -s add_header.py
```

Жмём | и запускаем game



## Chapter 3

# CI

We got a very old continious integration system (on OpenIndiana) and we want to upgrade it.

Please upgrade project serverMVC to use Docker.

Out checker works with your repo at `ssh://root@${YOUR_IP}/root/app.git` and for start deploy checker will use url:

`http://${YOUR_IP}:8080/hudson/job/serverMVC_docker/build?delay=0sec`

Moreover, checker expects Docker running on `${YOUR_IP}`.

Our `id_rsa.pub`: `ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDoGF7Dvs5H0aeMsfm9MMasUWY12rdphM410FJdJ aoAjGFA8X6PeKq06qEbZpbmXR0Yfrcwn1hRONAQ  
FHjHvtfp4ViVOtfmt1byubG9BaFRJe+L3+17MkAodYyrC93/jisk3xi/veAkuRFa4F7qUioBOuRYXE KSg4eF+tMouqbzKoM2O9vsAHRBRaIhV+yTiiDjN2UswzmQl4n4m/wRZ/OKISiewUzol  
i3Lpl5IdaNMiYdsi9D8Mgb7N2x4DKZKTXOVnHmMN79yL1u2WUlp3vhWAmz8Af4Sux7jh`

# Chapter 4

## netflow

Set up a netflow receiver on port 9996 and make a traffic billing.

For each user you should write bytes count and make it available via `http://<your-ip>/billing.html` which looks like:

```
<tr>
  <td>IP</td>
  <td>bytes count</td>
</tr>
```

Update period: 1 minute

Sort: by bytes DESC

Ставим flow-tools. Настраиваем. Теряем 3 часа, чтобы понять, что нам нужны только 220.123.31.0/24. Готово.

```
1 -w /var/flow/ -n 1339 -N 0 0/0/9996 -R /root/bin/flowrep.sh
```

Listing 2: flow-capture.conf

```
1 stat-report localnet
2   type ip-address
3   output
4   format ascii
5   options -header,-xheader,-totals
6   fields -flows,+octets,-packets,-duration
7
8 stat-definition localnet
9   report localnet
```

Listing 3: report.conf

```
1 #!/bin/bash
2 FILE="/var/www/flow/billing.html.new"
3 echo "<html><head></head><body><table>" > $FILE
4 flow-cat /var/flow/ft-* | flow-report -s /root/bin/report.conf -S localnet | \
5   grep -v '\#' | sort -r -n -k 2 -t ',' | grep "220.123.31." | \
6   awk -F',' '{print "<tr>\n  <td>\"$1\"</td>\n  <td>\"$2\"</td>\n</tr>"}' >> $FILE
7 echo "</table></body></html>" >> $FILE
8 mv /var/www/flow/billing.html.new /var/www/flow/billing.html
```

Listing 4: flowrep.sh

## Chapter 5

# Repo

We got a repository at /root/repo, but it doesn't work with youm < 3.0.0. Fix it and make aviable via <http://<ip>/repo>.

Сразу смотрим в ман по createrepo:

The older default was "sha", which is actually "sha1", however explicitly using "sha1" doesn't work on older (3.0.x) versions of yum, you need to specify "sha".

На всякий случай бэкапим. С дуру попробовал пересоздать repodata с указанием алгоритма sha, сломал. Восстановил, открыл руками repomd.xml и поменял тип и хэш-суммы.

```
cd /root/repo/repodata/

for i in *.bz2; do
    sha_old=$(sha256sum $i);
    sha_new=$(shasum $i);
    sed -i -e "s/$sha_old/$sha_new/g" repomd.xml;
done

for i in *.bz2; do
    bzcat $i > ${i%.bz2};
    sha_old=$(sha256sum $i);
    sha_new=$(shasum $i);
    sed -i -e "s/$sha_old/$sha_new/g" repomd.xml;
    rm -f ${i%.bz2};
done

sed -i -e "s/\\"sha256\\"/\\"sha\\"/g" repomd.xml
```

## Chapter 6

# Infected binary

Your image has been touched by a cracker who replaced one of standard system binaries with his own. We thought that the program contains some secret string and it will output it when properly executed. Please find that string.

Ну это совсем просто. Ищем изменённые бинарники (проверяем чексумму), затем смотрим в него.

```
apt-get install debsums -y
debsums -a -s
strings /usr/bin/\[ | more
```

Ага. Записать строчку в binary.txt и распарить по http. Готово.

Сделано быстрее всех

# Chapter 7

## DNS MITM

Let's suppose you're an administrator of large corporate network. You have a list of hosts which should be blocked according to the company policy. You decide to set up your DNS server in such a way that it acts as normal DNS server for all the hosts except the ones from this list. For those hosts, it returns the address of itself to all A queries. Later you plan to set up a special web server that displays the page about your company Internet restrictions on the same machine.

Now the task is to set up such DNS server. You should find the list of hosts in /root.

Сначала я подумал, что подменять запросы нужно будет для отдельных клиентских IP, и приготовился сделать 2 view в bind

Но оказалось, что блокировать нужно отдельные домены. ещё проще

```
# устанавливаем BIND и утилиту dig
apt-get install bind9 dnstools

# создаём часть конфига, добавляем в него блокируемые домены
# некоторые домены повторяются, поэтому uniq
cat /root/dns.hosts | sort | uniq | while read line; do
    echo "zone \"$line\" {" ;
    echo "    type master;" ;
    echo "    file \"/etc/bind/db.fake\";" ;
    echo "}";
done > /etc/bind/named.fake.conf

# часть результата:
root@postel:/etc/bind# head -n 8 /etc/bind/named.fake.conf
zone "12zeed.com" {
    type master;
    file "/etc/bind/db.fake";
};
zone "link.com" {
    type master;
    file "/etc/bind/db.fake";
};

# включаем его в основной конфиг /etc/bind/named.conf
include "/etc/bind/named.fake.conf";

# создаём файл зоны. один на всех, благодаря шаблону "@"
$TTL 86400
@      IN      SOA     postel.yandex.com.  postel.yandex.com (
                        2015042201
                        3H
                        15M
                        1W
                        1D )

@      IN      NS       postel
postel IN      A        10.0.1.51
@      IN      A        10.0.1.51
*      IN      CNAME    @

# в другом конфиге/etc/bind/named.conf.options включаем forwarders ( не обязательно, у нас уже есть есть zone "." type hint )
# и полное логирование ( а это вообще нежелательно, т.к. снижает производительность на порядки )

# создаём каталог для логов
mkdir /var/log/bind
chown -R bind /var/log/bind

# запускаем bind
service bind start

#проверяем:
```

```

root@postel:/etc/bind# dig +short ya.ru @localhost
213.180.193.3
213.180.204.3
93.158.134.3
root@postel:/etc/bind# dig +short $(head -n1 /root/dns.hosts) @localhost
10.0.1.51

```

```

1 options {
2     forwarders {
3         8.8.8.8;
4     };
5     allow-recursion {
6         127.0.0.1/8;
7         10.0.0.0/8;
8         ::1;
9     };
10 }
11 logging {
12     channel update_debug {
13         file "/var/log/bind/bind_upd.log";
14         severity debug 3;
15         print-category yes;
16         print-severity yes;
17         print-time yes;
18     };
19     channel security_info {
20         file "/var/log/bind/bind_sec.log";
21         severity info;
22         print-category yes;
23         print-severity yes;
24         print-time yes;
25     };
26
27     channel simple_log {
28         file "/var/log/bind/bind.log" versions 3 size 5m;
29         severity warning;
30         print-time yes;
31         print-severity yes;
32         print-category yes;
33     };
34
35     #channel query_log {
36     #     file "/var/log/bind/bind_query.log" versions 3 size 5m;
37     #     print-time yes;
38     # };
39
40     channel update_log {
41         file "/var/log/bind/bind_update.log" versions 3 size 5m;
42         print-time yes;
43     };
44
45     category update { update_debug; };
46     category security { security_info; };
47     category default { simple_log; };
48     category update { update_log; };
49     category update-security { update_log; };
50     #category queries { query_log; };
51 };

```

Listing 5: named.conf.options

P.S. когда описывал решение, обратил внимание что это решение не универсальное и даже не полностью выполняет задание

Что ж, баги проверяющего робота - это приятно ;)

## Chapter 8

# SVN

You have svn repository in /root/repo. Delete (like svn rm) all files which are greater than 5MB in all revisions and make them available via svn://ip/

Big file should be deleted only in the revision when it became >5MB.

К сожалению, мега-супер-пупер-крутой скрипт был утерян ввиду анализа Infected binary через gdb, так что вкратце алгоритм:

1. Ставим более свежие версии svn и svn-utils из wheezy-backports
2. Запускаем svnserve на /root/repo
3. Делаем локальный клон через svn co
4. Дальше скрипт:
  - Дампить все ревизии через svn dump (1 - полная, остальные с -incremental)
  - Смотреть какой файл превышает 5МиБ через svn ls -verbose
  - Добавлять к списку exclude
  - Прогонять через svndumpfilter
5. Получили 101 отфильтрованную ревизию
6. Теперь через svnadmin create создаём репозиторий и через svn load грузим фильтрованный дамп
7. Сервим новый реп
8. PROFIT!

# Chapter 9

## NFS

Make `nfs://10.10.10.11/dir` available as `http://<yourip>/nfs`. Use this credentials `user@YA.ROOT:password`. NFS server's host name is `localhost`.

Домен в логине - явный намёк на realm Kerberos'a

Где искать KDC ? Неизвестно. Предполагаем, что на том же 10.10.10.11

```
# ставим утилиты kerberos и nfs клиента
apt-get install krb5-user nfs-common

# добавляем в конфиг
[libdefaults]
default_realm = YA.ROOT
allow_weak_crypto = true

[realms]
YA.ROOT = {
    kdc = 10.10.10.11
    admin_server = 10.10.10.11
    default_domain = ya.root
}

[domain_realm]
.ya.root = YA.ROOT
ya.root = YA.ROOT

#проверяем
kinit -p user@YA.ROOT

#успешно получаем TGT, можно даже посмотреть его
klist

# Дальше был затык и несколько часов. Я использую nfs4 + kerberos дома уже лет 8 и глаз замылился.
# Всегда использовал sec=krb5i, а для этого нужен keytab на имя host/$client.$domain.$REALM.
# Keytab искали по обеим ВМ
# Пытался подобрать principal для keytab'a
# Даже зашёл на kadmin от имени того же user@YA.ROOT и пробовал искать и экспортировать principal'ы.
# Почему-то яндекс забыл закрыть доступ :)
# От безысходности экспортировал в keytab сам user@YA.ROOT
# Не прошло
# Зато оказалось, что при экспорте _меняется_пароль_. Ну ладно, kinit и kadmin можно выполнять и через keytab.
# Но т.к. . сервер был _общий_ на все команды, то я стал хакиром, который портил жизнь соперникам }:->
# Парни, я не специально. В запарке не сразу сообразил что сделал.
# Яндекс тоже опомнился не сразу, но чуть позже доступ к kadmin заблокировал
# Пробовал монтировать и без кейтаба, с ожидаемым эффектом
# mount -t nfs4 10.10.10.11:/dir /mnt/nfs -o sec=krb5i

# Потом lagrabee ВНЕЗАПНО сказал, что шара смонтировалась
# монтировать нужно было без sec=krb5i, т.е. без проверки сервера и клиента через KDC. Ну что сказать, глаз замылился
mount -t nfs4 10.10.10.11:/dir /mnt/nfs

# После этого каталог был срочно экспортирован через nginx, пользотель nginx тоже получил TGT

sudo -u www-data /bin/bash
$ kinit -p user@YA.ROOT

# /etc/nginx/sites-enabled/default:

server {
    listen 80;
    server_name 10.0.1.51 localhost postel postel.yandex.com postel.ya.root;

    location /nfs {
        alias /mnt/nfs;
    }

    access_log /var/log/nginx/postel_access.log;
```



```
    error_log /var/log/nginx/postel_error.log;
}

#можно проверить wget'ом
wget http://0.0.0.0/nfs/$(ls -l /mnt/nfs | head -n1 )
```

## Chapter 10

# Nginx Lua

We have inherited from previous admin several modules for nginx, you can find it out at `/etc/nginx/lua`. No documentation, no examples of configuration files. But we have some examples, how web-server on port 8000 must work:

- `/static/local/jquery.min.js` should return content of file `/var/www/static/jquery.min.js`
- `/static/local/<size>/dog.png` should return thumbnail of image `/var/www/static/dog.png`, with width and height limit equals to X. `<size>` is a verbal description of size ("small", "medium" and so on, full list is unknown). We have values for X for different sizes: 50, 100, 500, 1000, 2000.
- All requests to `/static/local/*` except requests for css- and js- files, should return error 403 unless user has a special authorization cookie
- Request to `/auth/local/jquery.min.js` should set authorization cookie with name "auth\_local/jquery.min.js", which is accepted by `/static/local/jquery.min.js`

This list is not complete! But it's all we have.

Don't forget to setup caching. It is said these modules support them.

```

1  #!/usr/bin/env python3
2  # ACHTUNG! THIS IS NOT HOW I USUALLY WRITE SCRIPTS!
3
4  # This supposed to be code in nginx+lua, but I found
5  # python more convenient for handling tricky URLs and
6  # image manipulations. Sorry :)
7
8  from bottle import route, get, run, template, \
9      static_file, request, response, HTTPError
10
11 import Image
12 import time
13 import re
14
15 SIZES = dict(xsmall=50, small=100, medium=500, large=1000, xlarge=2000)
16
17 @get('<uri:path>')
18 def static(uri):
19     # HERE IS WHERE YANDEX EXPETS CACHE TO BE
20     if uri.find('/static/remote/') != -1:
21         # quick and dirty way to parse URL, do not do in so in prod, sons
22         toks = uri.split('/')
23         size = toks[3]
24         fname = toks[-1]
25         if fname.endswith('png'):
26             fname = 'dog.png'
27         else:
28             fname = 'turtle.jpg'
29         uri = '/static/local/%s/%s' % (size, fname)
30         return static_file(uri, root='.')
31
32 # AUTH REQUEST, LET'S SET COOKIE
33 elif uri.startswith('/auth'):
34     fname = uri.split('/')[2:]
35     fname = "/" .join(fname)
36     # bits and pieces from reverse engineered lua code
37     response.set_header('Set-Cookie', '%s=%s; path=/; expires=%s' % ('auth_' + fname, 'xxx', int(time.time())+24*3600 ))
38     return "" # doesn't really matter what to return
39
40 # STATIC FILES!
41 elif uri.endswith('.css') or uri.endswith('.js'):
42     if uri == "/auth/local/jquery.min.js":
43         response.set_cookie("auth_local", "jquery.min.js")
44         response.set_header('Set-Cookie', 'auth_local/jquery.min.js')
45         uri = "/static/local/jquery.min.js"
46     # static_files() screws headers, so I use open() :(((
47     return open('.') + uri, 'rb') # DO NOT DO THIS IN PROD!!!
48
49 # IMAGES
50 elif uri.endswith(".png") or uri.endswith('.jpg'):
51     # check all cookies if there is auth cookie....
52     for c in request.cookies:
53         if c.startswith('auth_'):
54             # oh no, what a dirty code...
55             uri = '/static/' + c.split('auth_')[1]
56             break
57     else:
58         # no auth cookie, get out!
59         raise HTTPError(403)
60
61     # got auth cookie, let's resize img and put in on disk
62     r = re.match('/static/local/([a-zA-Z]+)/(.+)', uri)
63     vsize, fname = r.groups()
64     img = Image.open('static/local/' + fname)
65     size = SIZES[vsize]
66     img = img.resize((size,size))
67     img.save('.') + uri
68     return static_file(uri, root='.')
69
70 # all other URLs are forbidden (doesn't really matter)
71 raise HTTPError(404)
72
73
74 run(host='0.0.0.0', port=8000, debug=True, reloader=True)

```

Listing 6: nginx\_lua.py

Part III

Итоги

6 место. Огромное спасибо tazhate, madrouter, realloc, lumi, octocat, delirium, anonymous\_sama, trofk, feofan, tailgunner, всем кого забыл, ну и мне %)

Backup	HTTPS MITM	CI	netflow	Repo	Infected binary	DNS MITM	SVN	NFS	Nginx Lua
01:29:18	04:27:47	07:01:54	05:25:27	01:39:15	00:18:49	01:27:47	03:29:41	04:52:19	07:06:34